A hybrid and adaptive approach to humanoid locomotion: blending rhytmic primitives and feet placement strategies

José Pontes

Sorbonne Université

Universidade do Minho

École doctorale Informatique, Télécommunications et Électronique (Paris)

Institut Systèmes Intelligents et de Robotique (ISIR - - CNRS UMR 7222)

A hybrid and adaptive approach to humanoid locomotion

blending rhythmic primitives and feet placement strategies

Par José Pontes

Thèse de doctorat de Robotique

Dirigée par Stéphane Doncieux, Cristina Santos, et Vincent Padois

Présentée et soutenue publiquement le 18 Mars 2019

Devant un jury composé de :

M. A. Ijspeert	Professeur, Université de Southern California	Rapporteur
M. A. Pereira	Maître de Conférences, Université de Aveiro	Rapporteur
M. O. Sigaud	Professeur, Sorbonne Université	Examinateur
M. V. Alves	Maître de Conférences, Université du Minho	Examinateur
M. S. Doncieux	Professeur, Sorbonne Université	Directeur de thèse
M. C. Santos	Maître de Conférences, Université du Minho	Co-encadrant

Acknowledgements

I would like to thank my advisors, Cristina Santos, Stéphane Doncieux, and Vincent Padois, for their invaluable advice, dedication, and patient throughout the whole project.

I would also like to acknowledge the support from the research units that supported me both in Portugal (centro ALGORITMI and CMEMS), and France (ISIR), providing the infrastructure and material support needed.

The presented work was possible thanks to the support by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through the PhD grant with the reference SFRH/BD/86270/2012, and the Programa de Ações Universitárias Integradas Luso-francesas (PAULIF).

Abstract

This thesis presents research developed towards adaptable and effective optimization of controllers for humanoid robots' locomotion. It addresses this goal by combining features of humanoid locomotion and the impact of different environments in its dynamics, along with mathematical optimization and statistical analysis techniques.

Controllers for the locomotion of bipedal robots often face challenges regarding their optimization towards different objectives and different environments. We propose an architecture that uses the information gathered in an optimization/exploration phase to adapt to a terrain with partially unknown characteristics. In the exploration phase virtual simulations are used to optimize the parameters of the controller in different terrains. The results of these optimizations are used to identify the unknown terrain characteristics, and these values are used to select the best parameters for this particular environment. The approach was tested in the simulations of an iCub robot on terrains with variable friction, and of the DARwIn-OP robot in ramps with varying slopes.

This work brings contributions to the problem of choosing the values for the open parameters of biped locomotion controllers in various situations. Specific issues that are covered relate to: 1) establishing an optimization framework that can be applied to any controller with open parameters, and which results in both safe and well performing behaviors; 2) finding locomotion behaviors that are effective in multiple and distinct environments; 3) adapting the locomotion to environments with varying characteristics, modeled in previous optimizations.

Résumé

Cette thèse présente des recherches développées pour une optimisation adaptative et efficace des contrôleurs de locomotion des robots humanoïdes. Elle répond à cet objectif en combinant les caractéristiques de la locomotion humanoïde et l'impact de différents environnements dans sa dynamique, ainsi que des techniques d'optimisation mathématique et d'analyse statistique.

Les contrôleurs pour la locomotion des robots bipèdes font souvent face à des défis concernant leur optimisation vers différents objectifs et environnements. Nous proposons une architecture qui utilise les informations recueillies dans une phase d'optimisation / exploration pour s'adapter à un terrain avec des caractéristiques partiellement inconnues. Dans la phase d'exploration virtuelle des simulations sont utilisées pour optimiser les paramètres du contrôleur dans différents terrains. Les résultats de ces optimisations servent à identifier les caractéristiques inconnues du terrain et, en conséquence, à sélectionner les meilleurs valeurs des paramètres du contrôleur pour cet environnement. L'approche a été testée dans les simulations d'un robot iCub sur terrains avec un frottement variable, et du robot DARwIn-OP dans des terrains avec des pentes différentes.

Ce travail s'intéresse donc au problème du choix des valeurs des paramètres des contrôleurs de locomotion bipède dans diverses situations. Les contributions apportées consistent à : 1) établir un cadre d'optimisation qui peut être appliqué à n'importe quel contrôleur avec des paramètres ouverts, et qui se traduit par des comportements à la fois sécurisés et performants ; 2) trouver des comportements de locomotion efficaces dans des environnements multiples et distincts ; 3) adapter la locomotion à des environnements aux caractéristiques variables, modélisés dans optimisations précédentes.

Resumo

Esta tese apresenta pesquisa desenvolvida no sentido de chegar a uma otimização adaptável e eficaz de controladores para a locomoção de robots humanoides. Para atingir este objetivo, são combinadas características da locomoção e o impacto que diferentes ambientes têm nas sua dinâmicas, em conjunto com técnicas de otimização matemática e análise estatística.

Controladores da locomoção de robots bípedes frequentemente enfrentam desafios relacionados com a otimização relativa a diferentes objetivos e ambientes. Aqui propõe-se uma arquitetura que usa a informação recolhida numa fase de otimização/exploração para adaptar a marcha a um terreno com características parcialmente desconhecidas. Na fase de exploração, simulações virtuais são usadas para otimizar os parâmetros do controlador em terrenos diferentes. Os resultados dessas otimizações são usados para identificar as características do ambiente que são desconhecidas, e esses valores são usados para selecionar os melhores parâmetros para esse terreno. Esta abordagem foi testada em simulações de um robot iCub em terrenos com fricção variável, e de um robot DARwIn-OP em rampas com diferentes declives.

Este trabalho oferece contribuições relativas ao problema de escolher valores para os parâmetros de controladores da locomoção de bípedes em várias situações. Especificamente, cobrem-se questões relacionadas com: 1) estabelecer uma plataforma de otimização que pode ser aplicado a qualquer controlador com parâmetros por definir, e que resulta em marchas seguras, e com bom desempenho; 2) encontrar marchas que são eficazes em ambientes múltiplos e distintos; 3) adaptar a locomoção a terrenos com características variáveis, modeladas em otimizações prévias.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This document contains a detailed description of the work developed by the author in his doctoral studies while integrated in the Adaptive System Behavior Group at Universidade do Minho and in the Institut des Systèmes Intelligents et de Robotique at Université Pierre et Marie Curie.

The work addresses the problem of optimizing bipedal robot locomotion controllers, so that they can adapt to different user specifications, like being faster, or more secure, and also to different traversed terrains. It does so by incorporating approaches from multi-objective optimization and sensitivity analysis in a framework that was tested in locomotion control systems based on Central Pattern Generators, and on model-based reactive whole-body control formulated as a constrained convex optimization problem.

1.1 Motivation

Locomotion is an essential skill for humanoid robots, and designing its controllers is difficult (Vukobratovic et al., 1990). Humanoids are under-actuated, free floating systems, and walking requires creating and breaking unilateral contact between the feet and the ground (Brogliato, 1996). These characteristics render the dynamics of locomotion hybrid and nonlinear. Moreover, bipedal robots usually have a high number of degrees of freedom (DoFs), which allow different body parts to move in relation to each other. This can add complexity to the task of walking, since it does not uniquely specify how the limbs must be coordinated in order to achieve the desired displacement of the robot's center of mass (CoM) (Westervelt et al., 2007). The task is, then, a problem with many solutions, and unfeasible solutions (i.e., that make the robot trip and fall) can be common (Westervelt et al., 2007). In this context, finding a solution that is "good", even if it is not optimal, may be difficult without proper tools to separate the appropriate solutions from the rest (Milutinović and Rosen, 2013). The variety of possible behaviors can also lead to the need to choose the best performing ones, exacerbated if there are different performance requirements at different times, or from different users.

Robot locomotion control approaches contain parameters that can be tuned in

order to optimize the locomotion towards certain goals. These goals can be varied, with the robot not falling usually being the most important one, and its speed, energy consumption, and stability factors being other good examples of what a user may require from the locomotion of the biped. On top of the locomotion having to respect these objectives, the terrain in which the robot walks can present certain characteristics that make it harder to find the best solution for the walking task (Afshar and Ren, 2012; Voloshina et al., 2013; Kajita and Tani, 1991; McCown-McClintick and Moskowitz, 1998). Walking in a floor with especially high or low roughness, or with a slope, for instance, can make it harder for the robot to walk. Not only that, but simply changing the conditions, even when that makes the task easier, usually implies changing the controller parameters in order to adapt the solution to one that is adequate for the new terrain.

Given all these variables that can change the relationship between the controller's parameters and the resulting locomotion dynamics, as well as changing requirements from the human users and the complexity of these dynamics, tuning these controllers can be something hard or impossible to achieve with analytical studies based on mathematical models of the locomotion's dynamics (Holmes et al., 2006). Modeling is further complicated by the need to take into account the surrounding environment, which can be hard to observe and analyze. The alternatives to this analytical approach are to tune them manually, through experimentation, or to resort to derivative-free optimization techniques that do not require accurate models of the dynamics. Both these approaches rely on running the entire system (either through simulations with a model or in the real robot) to evaluate specific groups of input values. Manual tuning is prohibitive in terms of time for any controller architecture that contains parameters that can be set for a continuous number of values, or simply have a high number of open parameters, since it results

1.2 Locomotion controllers adaptation

There are many works that deal with controller optimization in general, and adaptation to environment changes in particular. The question is, is there a framework that deals with this problems in a unified, encompassing way, while needing the least amount of expert users input for different situations?

As a couple of examples, both the model-based control from Herdt, Perrin, and Wieber (2010), and the central pattern generator based control from Liu, Wang, and Chen (2013) can directly change values like the length or height of each step. This can be used along with sensory information to adapt to specific obstacles of uneven terrain at runtime, such as the approach used by Gay (2014), where sensory information is used to modify the dynamics of a controller.

Kimura, Fukuoka, and Cohen (2007) and Wang et al. (2010) present broader strategies, looking to adapt to different terrain changes. The former design a central pattern generators based controller to comply with specific necessary conditions for stable dynamic walking set by the authors. This makes the approach specific to the controller

developed. The approach by Wang et al. (2010), representing variables such as external forces and control torques as probability distributions, is a computationally complex one that was only applied to a simple four state PD controller.

A more general approach can be found in Cully et al. (2015), which creates a map of diverse sets of parameters for a given locomotion controller, and uses it t adapt to environment or robot changes during locomotion. It can be used with any controller with open parameters, and the adaptation can be done to different context changes. This framework has problems regarding flexibility, such as using features that are specific to a robot model, and obtaining a map of solutions that is diverse by design, which means the optimization might spend too much time with low performing solutions.

This thesis aims for a framework that is flexible in terms of optimizing for any locomotion controller applied to any bipedal robot. This optimization should be able to support an adaptation to any context that can be codified as a single valued variable. The framework will focus on automating each step as well as possible, using the information from the optimizations to make decisions on what controller parameters and locomotion features should be used in different situations. According to the research done, there is currently no framework that achieves this objectives in their entirety.

1.3 Goals and methods

The main goal of this work is to devise a general solution for the adaptation of biped locomotion controllers to different environments and different user goals. We aimed to develop a framework that:

- Can adapt the locomotion to different terrains (*e.g.* different floor frictions or slopes).
- Can adapt the locomotion to different user defined objectives (*e.g.* maximize speed and/or minimize joint torque).
- Is able to make these adaptations in a minimal amount of time.
- Is not specific to a controller architecture, or a robot model.
- Requires a minimal amount of robot sensors.

The proposed method achieves the required adaptations by analyzing information related to the relationship between the parameters of a locomotion controller and some of the locomotion features. This data is acquired by running several simulations (in the order of several thousands) of the locomotion task of the robot, optimizing the parameters of the controller towards the desired features (*e.g.* speed).

In order to conduct the locomotion adaptations in a reasonable amount of time, the method is separated in **two distinct phases**: an **exploration (or optimization)**

phase, which is longer and done in preparation to the **adaptation phase**, which should be quick, taking no more than a couple of minutes.

In the exploration phase thousands of simulations are performed, resulting in a large amount of information that can be used in the next phase. These simulations are guided by an optimization algorithm. This algorithm should be one that does not rely on the derivative of the system's mathematical function, since there is no access to it, which typically implies the need for thousands of trials, leading to the necessity of using a simulated environment to conduct them. This phase also includes a sensitivity analysis to study how well certain sets of parameters perform in different terrains, and a correlation analysis to inform on which parameters are more relevant to our intended locomotion outputs.

In the adaptation phase the robot is put walking in an unknown terrain it has to adapt to. The locomotion features observed in that terrain are compared to the results from the exploration phase, and a set of parameters that results in an optimal or near-optimal behavior is selected. A solution that shows a balance between locomotion features diversity (making a distinction between different terrains easier) and locomotion safety (choosing a solution that ensures the robot does not fall) is chosen for the first trial in the unknown terrain. In the second trial, after making an identification of the terrain parameter, the solution chosen is one that is optimal in this terrain and meets the user performance requirements (e.g. the highest possible speed).

1.4 Contributions

This work's contributions relate to the analysis of the effect of changes in the parameters of different controllers for the locomotion of bipedal robots, as well as the adaptation capabilities of those controllers to different situations. It addresses the adaptation to different terrains and different user requirements (relative to the locomotion features).

The effects of changing a controller's parameters are tested for virtual models of the iCub (Sandini, Metta, and Vernon, 2007) and the DARwIn-OP (Ha et al., 2011) robots, as well as a virtual manikin. Optimization procedures in different environments and robot structures provide insight to how these changes affect the locomotion task. For the virtual manikin, the outcome of changes in its height and weight, while keeping its kinematic structure, is analyzed. The iCub and DARwIn-OP virtual models are tested in floors with varying friction and slope, respectively.

A correlation analysis is proposed, giving insight on the impact that the parameters of a controller have on the locomotion. It analyzes the statistical covariance between these parameters and the locomotion features. This information is then used to tune the optimization procedure by reducing the number of parameters used. Redundant parameters, and parameters that have low impact in the output, are removed from the process.

A sensitivity analysis allows for further exploration of the results of the optimizations, showing how effective some of the solutions are in different conditions. This process consists in testing solutions in multiple contexts, and analyzing how effective

they are in terrains they were not optimized for.

An adaptation process for a locomotion controller under different terrain conditions is outlined. This process also enables adaptation to different user requirements relative to the optimized locomotion features (*e.g.* average speed, joints torque output). It does so by identifying the value of a modeled variable of the terrain, and then choosing the best set of controller parameters for the environment. It makes these decisions by considering the information gathered from the optimizations and sensibility analysis from the previous experiments.

1.5 Outline

After this introduction, Chapter 2 delves into humanoid robots's locomotion control, and the problems surrounding its optimization and adaptation to different conditions. Chapter 3 presents additional background related to the mathematical optimization of a system, and, more specifically, evolutionary algorithms, as well as a detailed presentation of the nondominated sorting genetic algorithm II (NSGA-II), which was used in the experiments included further into the document. The first stage of the thesis overall strategy is presented in Chapter 4, which contains an approach for controller optimization that is applicable to multiple robots and control strategies. This chapter also details the possibles components of this framework, as well as proposing a sensitivity analysis, and a correlation analysis, that provide insight into the process, and help fine tune it. Chapter 5 shows the setups and results of the application of this optimization framework to different robot models, control architectures, and varying terrain conditions. The second stage of the overall strategy — the adaptation framework — is presented in Chapter 6, which, building on the previous optimization, details each function used in the process of identifying the locomotion context, and adapting to it. This adaptation framework was applied to some of the same setups from the previous experiments, with the methodology and results shown in Chapter 7. Finally, Chapter 8 presents the main conclusions from the thesis, with focus on the results obtained in the main experiments, as well as perspectives on future improvements for this work.

1.6 List of publications

1.6.1 Conference papers

- José Pontes, Stéphane Doncieux, Cristina Santos, and Vincent Padois, *An adaptive approach to humanoid locomotion*, Proceedings of the 19th International Conference on CLAWAR 2016.

Chapter 2

Humanoid robot locomotion control

Bipedal locomotion is a complex task which control needs to take into account multiple restrictions and performance criteria. Locomotion controllers in biped robots often face the challenges of having to adapt themselves to the dynamics of the locomotion, the reality of the environment surrounding the robot, and the user requirements for the task. Some controllers are able to take care of some of these aspects to a greater or less degree, but not all of them in a general way.

The goal of this work is to build a framework that can be used with any humanoid locomotion controller with open parameters (*i.e.*, parameters whose values are not set) and that can be used to optimize them towards user requirements, as well as to adapt them to changes related to the locomotion process, such has changes in the environment, or in the robot itself.

This chapter outlines the basic characteristics of humanoid locomotion in order to give context to the most common approaches to its control. In discussing these, we will highlight how the existence of open parameters in these controllers, and many others, leads to the possibility of tuning these parameters for better outcomes in an optimization process. Afterwards, comes a discussion of some approaches to that process, and the chapter finishes with a discussion of adaptation to changes in the locomotion task that are external to the controller (more commonly, terrain changes).

2.1 Humanoid robots locomotion

There are advantages to an humanoid robot mimicking the human body in appearance and functionality. Humanoid robots can more easily work in environments and with equipments designed for humans (Kajita et al., 2014), and there is increased empathy and understanding for closer interaction between humans and robots (Meng and Lee, 2009). Moreover, there is the postulation that natural structures and processes were evolved towards being closer to optimal in the context of what is possible with a given structure (Alexander, 1996), and therefore desirable to replicate.

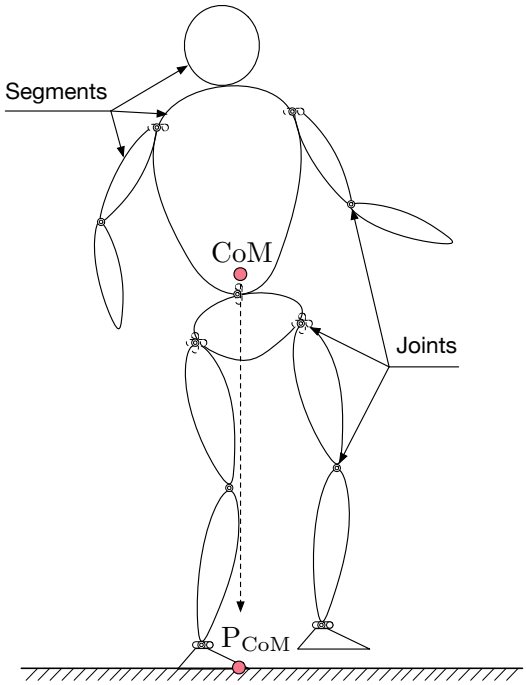


Figure 2.1: Abstract link-segment of a biped locomotion system. Approximations to the system's Center of mass of a robotic system (CoM), and its projection on the ground (P_{CoM}) are also represented.

Fully understanding and replicating a walking motion in a stable and optimal fashion is a challenge for people in multiple fields, spanning biology, physiology, medicine, mathematics, and engineering, and its optimization is subject to multiple criteria (Vukobratovic et al., 1990). Mechanical complexity observed in biological systems is usually replicated in robots by individually fabricated parts, which are costly in mass-scale production. Humans share this mechanical complexity, along with biological energy storage systems that can achieve torque, response times, and conversion efficiency that exceed the man-made robotic systems with a similar scale (Siegwart, Nourbakhsh, and Scaramuzza, 2011). These production related problems are accentuated by the nature of an underactuated system, and the associated nonlinear multibody dynamics. As such, humanoid locomotion is still unable to match the robustness and finesse of biological systems.

Because of the challenges in replicating the locomotion of humans, the design of the humanoid and its control system is usually done with resort to accurate kinematic and dynamic models. These are usually based on a link-segment model in which parts of the body are represented as rigid segments of constant length, with their mass concentrated at their CoM, and are linked by one or more hinge joints. A basic representation of a humanoid mechanism of this type can be seen in Figure 2.1. Humanoid locomotion systems have a high number of DoFs, which by itself causes complex dynamics and

coordinate frame handling. The lack of fixed frames of reference causes these to be underactuated systems where their interactions with the environment lead to a conversion of internal joint forces to external reaction forces (Vukobratovic et al., 1990). This external reaction forces are observed in the contact between the feet and the ground, which is essential for walking since it can cause the body's relative position to be changed (Vukobratovic and Borovac, 2004).

When walking, a humanoid comes in contact with the ground at multiple points, and these are broken and then recovered in order to generate movement (Wieber, 2002). This brings versatility to overcome obstacles, but also instability that requires the robot to be stabilized. The robot can collect information to use in that stabilization, using sensors that provide feedback about the robot itself, such as gyroscopes and accelerometers to find the position and the orientation of the robot, and force/torque sensors to measure the contact forces and torques between the feet and the floor (Kajita et al., 2014), or sensors that provide feedback of the robot environment, like vision or sound sensors.

2.1.1 Characteristics of humanoid locomotion

Humans adjust the locomotion pattern for the limbs — the gait — to adapt to different walking motions under different circumstances, such as changes in terrain, and to desired behaviors, including, but not limited to, a target walking velocity (Alexander, 1984). When describing the gait, a stride is a cyclical movement which is usually divided into the stance and swing phases. In the stance phase the foot is on the ground and can be used as a pivot to push the body forward, while maintaining an upright posture. In the swing phase the foot comes off the ground to enable the contrary leg to advance and its foot to enter its stance phase. There is a period between right after the heel strike of one of the feet and the toe off of the contrary foot where both feet are on the ground, consisting in a double support phase (Morecki, 1997). A visualization of these phases can be seen in Figure 2.2. In the double support phase the locomotion mechanism is a closed kinematic chain, because both ends of the chain (the feet) are in contact with the ground, while in the single support phase one of the ends has no anchor, making it a open kinematic chain (Vukobratovic and Borovac, 2004).

The human gait can be quantified with the stride (or step) frequency (the number of strides taken in unit time), the stride length (the distance traveled in a stride), the duty factor of a foot (the fraction of time for which the foot is on the ground), and the relative phase of a foot (the stage of the stride at which the heel strike occurs).

When a humanoid robot is in single support phase, its dynamics can be largely represented by an inverted pendulum connecting the support foot to the CoM of the robot (Kajita et al., 2001), referred to as the linear inverted pendulum model (LIMP), and represented in Figure 2.3. In this analogy, the kinetic energy of a stiff inverted pendulum being traded in potential energy, and sequentially back into kinetic energy, is akin to how energy is expended and transformed during a step (Kuo, Donelan, and Ruina, 2005). This simplification allows the design of a controller with limited information about its dynamics, such as only knowing the location of its CoM (Kajita

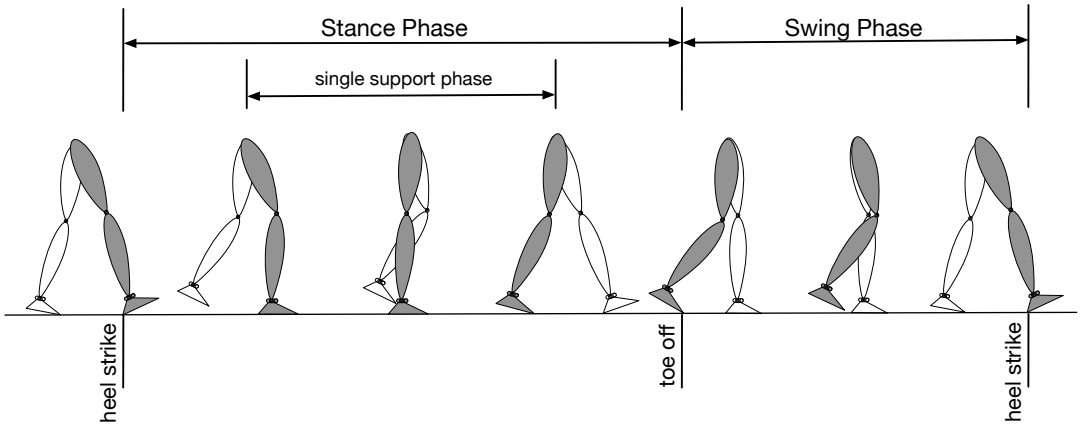


Figure 2.2: Phases of the human locomotion. The stretch of the stance phase that does not consist of the single support phase is called the double support phase.

et al., 2001).

2.1.2 Statically and dynamically balanced locomotion

Balance in bipedal locomotion can be divided in static balance and dynamic balance. Static balance is normally characterized by small velocities and accelerations, and is achieved if the system's projection of the CoM on the ground is kept in the supporting area of the feet. During static balance, the projection of the CoM in the ground corresponds to the center of pressure (CoP) — the point on the ground where all the ground reaction forces of the system act. Static balance is so referred to because the robot can stop at any instant and keep its balance. Dynamic balance is only maintained with continuous movement, and stopping without taking into account the body's equilibrium can result in a high risk of falling. In dynamic balance the projection of the CoM can be placed outside the supporting area of the feet, as long as the CoP of the mechanism is kept inside it (Nwokah and Hurmuzlu, 2002). The supporting area of the feet is the support polygon formed by the feet in contact with the ground, which is a convex hull of the supporting points, as exemplified in Figure 2.4.

The zero moment point (ZMP) is a concept introduced in the context of preserving the dynamic balance in legged locomotion (Vukobratovic and Borovac, 2004), and which is related to the CoP. The ZMP is termed as "the point where the influence of all forces acting on the mechanism can be replaced by one single force" (Vukobratovic et al., 1990; Vukobratovic and Borovac, 2004). In the single support phase, the dynamic reaction force and moment produced at the contact of the foot with the ground exist as a result of the moment and forces produced by the rest of the mechanism. If the point where this reaction occurs does not produce any moment in the horizontal direction, the body will not rotate around it, something which could cause the robot to fall. This is equivalent to the point where the total horizontal inertia and gravity forces equal

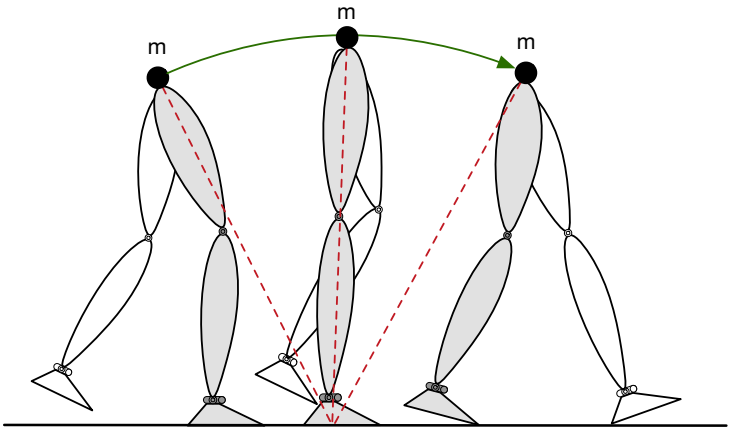


Figure 2.3: Representation of the linear inverted pendulum model for a humanoid. The support foot displacement (red dashed line) relative to the CoM (black circle labeled "m") takes the approximate path (green arrow) of a pendulum.

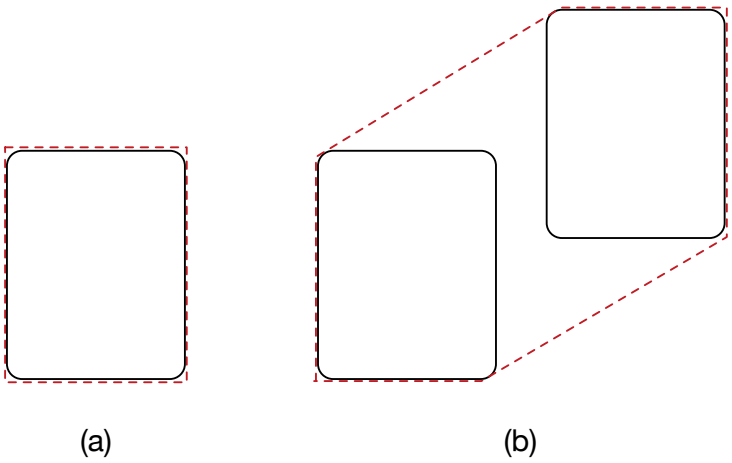


Figure 2.4: Representation of the support polygon of a biped mechanism in different support circumstances. Solid black rectangles represent the feet, while dashed red lines represent the support polygons. In the case of a single support phase (a), the support polygon is the area of the supporting foot, while in the double support phase (b) it is represented by the smallest convex set including all contact points (*i.e.* the convex hull).

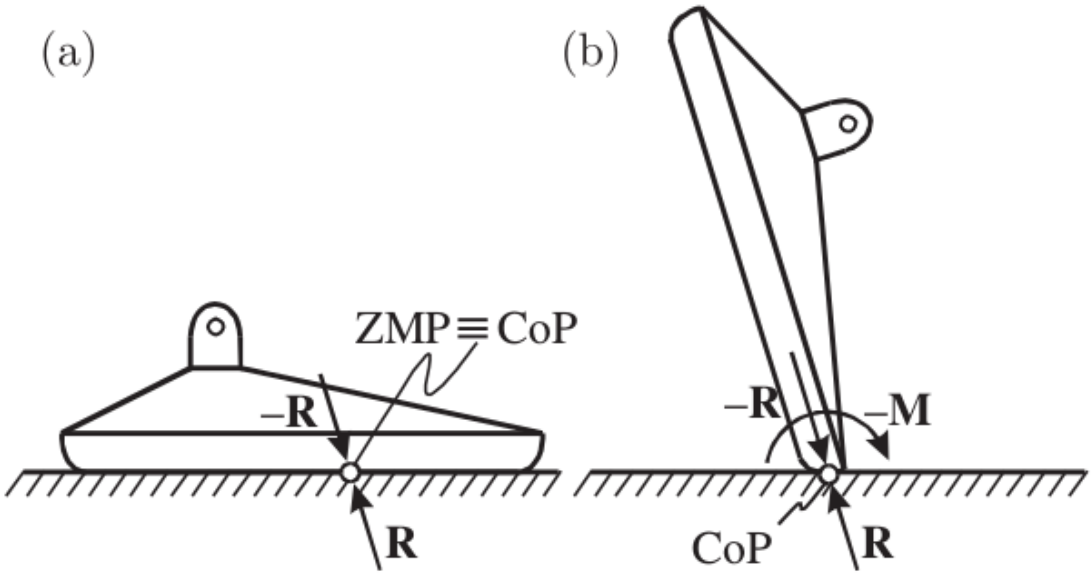


Figure 2.5: Relations between ZMP and CoP for a robot's foot in locomotion: a) dynamically balanced gait where the reaction forces R are of the same magnitude; b) unbalanced gait where the point where the sum of the moments is zero is outside the support polygon, which means there is not a ZMP for the robot, causing it to rotate about the foot edge with moment M and fall. Image adapted from Vukobratovic and Borovac (2004).

zero. For these reasons, the point was called the zero moment point. If the force acting at the CoP balances all forces acting on the mechanism in motion that point is also the ZMP, which means the CoP and the ZMP coincide on a dynamically balanced gait. When the balance does not exist, the ZMP also does not exist (*i.e.* the CoP is outside the support area) and the mechanism collapses around the foot edge. This relationship is illustrated in Figure 2.5.

The ZMP criterion, in its original state, is restricted to locomotion in flat ground, and unbounded tangential friction forces between the feet and the ground, which translates to a flat friction cone (Dai and Tedrake, 2016).

2.2 Biped locomotion control strategies

The problem of the control of a biped robot can be defined as choosing the proper inputs to its joints such that the system behaves in the desired fashion. One way to separate approaches to this problem is relative to the amount of information they use regarding the dynamic model of the system. One category of approaches uses precise information regarding dynamic information about the system, including the mass, the location of center of mass, and the inertia of each segment, as well as the

whole kinematic structure, which details how each segment is connected by a number of joints to other segments, forming a tree. The other category of this division uses little or no information about the system, such as using only its center of mass, or total momentum (Kajita et al., 2001).

Sections 2.2.1 and 2.2.2 refer to the most prevailing control strategies in these two categories of approaches, and highlight their general drawbacks regarding open parameters and consequent optimization towards certain goals. Later sections analyze works that tackle these optimization possibilities.

2.2.1 Control based on dynamic models

This type of control takes into account the dynamics of the robot, and its interactions with its environment during locomotion. In this, the robot is considered a multi-body system modeled with rigid bodies, and applies bounded torque on each joint. It follows Euler-Lagrange motion equations (Salini, Padois, and Bidaud, 2011),

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + N(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q}) + J_\chi(\boldsymbol{q})^\top \chi, \tag{2.1}$$

that relates the joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$, and accelerations $\ddot{\boldsymbol{q}}$, to the mass matrix $M(\boldsymbol{q})$, the nonlinear effects matrix $N(\boldsymbol{q}, \dot{\boldsymbol{q}})$, the gravity forces vector $\boldsymbol{g}(\boldsymbol{q})$ and the generalized wrench Jacobian $J_\chi(\boldsymbol{q})$, which describes how external forces affect the system in motion. χ is called the action variable, and composed by the vector of contact forces, and the vector of torque inputs, $\chi = [\mathbf{w}_c^\top, \boldsymbol{\tau}^\top]$. The motion equation can be used to optimize the system towards the torque inputs, the joint accelerations, or the contact forces (*e.g.* minimizing the accelerations).

A well-known example of control that uses a dynamic model of the controlled system is Kajita's preview control (Kajita et al., 2003). It uses a simplified version of the locomotion's dynamics, and not the full whole body dynamics of the system. In preview control the goal is to have an output that tracks a reference signal, in this case a trajectory for the ZMP of the system. In this work, they first started by modeling the dynamics of the locomotion of a biped robot as a 3D linear inverted pendulum. This model's purpose is to offer preview of the dynamics of the system that is accurate enough to enable locomotion, specifically the position of the CoM given the position of the ZMP, while being simple enough to be reduced to a handful of computationally inexpensive calculations. The system then generates a CoM trajectory such that the acceleration of the CoM throughout the locomotion is minimized and the resulting ZMP follows a given reference trajectory as closely as possible. This trajectory is then translated into a walking pattern by solving a problem of inverse kinematics. This approach has the constraint that the footsteps are fixed and impossible to change. These footsteps need to be predefined by the user, who needs to have enough knowledge about the environment to define a trajectory that results in a stable gait, and has no way to adapt to new environments automatically.

Expanding on Kajita's work, Wieber (2006) proposed what he called a linear model predictive control (LMPC) scheme, improving on the original ZMP preview control.

The ZMP equations from the previous work are formalized as a quadratic program (QP) — a mathematical optimization problem — that, given the tracking of a ZMP reference trajectory, minimizes the jerks (the derivative of the acceleration) of the CoM of the robot. The idea of Wieber's LMPC approach is to execute only a small part of the trajectory, and then recompute a new trajectory taking into account the current state of the CoM, allowing therefore for some feedback. The author also added a constraint to the QP that restricts the reference points of the ZMP to always stay a certain margin inside the convex hull of the two feet. This scheme was showed to be able to produce a CoM trajectory and adapt it in the middle of the locomotion after having a mass that corresponds to 33% of the total mass of the robot hit its trunk. It was not showed, however, how it would respond to variations of that perturbation (an obstacle for the feet would be more likely and more impactful), and, more importantly for our work, if this adaptation allows the robot to automatically adapt to different floor conditions. The main issue this scheme would have with this kind of adaptation is the fact that the output of the QP is a trajectory for the CoM: the joint commands are obtained through inverse kinematics, and if a perturbation to the environment changes the way the gait of the robot translates into CoM positions there is no way in this predictive control to account for this.

Later, Diedam et al. (2008) kept building on the LPMC scheme, making it so the positions of the feet do not have to be decided beforehand by a step planner, and are instead decided by simply adding new variables to the QP that correspond to the foot steps occurring over the prediction horizon. They propose to still use a step planner based on inverse kinematics, but keep complete freedom in the final choice of the step positions according to the robot stability and mechanical limits. When allowing for this freedom of choosing step positions, one also needs to make sure they will not lead to motions impossible to realize because of geometric and kinematic constraints of the robot (like leg length or joint limits), which was solved by adding inequality constraints to the QP. The controller's ability to adapt was tested in a similar way to the experiments done in Wieber (2006), but managing to recuperate from stronger perturbations.

Herdt et al. (2010) further improved on this motion generation scheme to allow it to generate stable walking motions without the use of predefined foot steps. A reference speed is given to the controller, and according to this speed and the current state of the robot a foot step placement is decided. Since the position of the CoP no longer follows a reference trajectory, the feasibility of the motion is obtained by constraining this position to lie in the middle of the feet positions decided by the algorithm. Later Herdt, Perrin, and Wieber (2010) improved the model predictive control (MPC) by adding an algorithm for the control of orientations of the feet and the trunk, allowing the robot to turn in a safe way, and also adding polygonal constraints on the positions of the computed feet positions to improve its reliability.

The work from Salini, Padois, and Bidaud (2011) and Salini (2012) uses the same Linear Quadratic Programming (LQP) optimization approach as the MPC, but it organizes the problem's constraints in a hierarchy, and treats them as different tasks to be

performed with an overall objective. These tasks consists in movements related to the whole body control of the robot. For the purpose of locomotion, it uses the trajectory tracking from a version of the ZMP Preview Control (Wieber, 2006) as the one of its tasks, and adds additional constraints that take into account physical laws of motion, the robot 's actuation limits and physical properties of the robot's environment. This hierarchical organization provides the ability to numerically change the priority of a postural task or the trajectory tracking, for example.

The changes to MPC introduced by Herdt et al. (2010) to make it so it only requires the reference speed as an input (and the dynamic, kinematic, and physical information of the robot controlled), causes the steps positions to be automatically selected with mostly stability in mind. This sometimes results in large oscillations in the system's forward speed in order to maintain the target mean velocity (Herdt, Perrin, and Wieber, 2010). It also translates in a lack of flexibility in choosing characteristics of the locomotion like the length, frequency, and width of each step, as well as the duty factor of the feet. This, in turn, can lead to having less potential to explore diverse behaviors provided by the controller. These behaviors could result in more optimal constant speeds or better energy efficiency, for example, with high energy costs being pointed as a functional shortcoming of MPC in general (Torricelli et al., 2016).

Because of this potential for exploring diverse behaviors, and because MPC is a robust and well regarded control system, it was chosen to be used in this project's work in parameter tuning and locomotion behavior exploration and optimization, leading then into terrain adaptation. The version of MPC used was that from Wieber (2006), implemented by Salini, Padois, and Bidaud (2011), since there was easy access to it and this specific implementation provides control over various parameters of the locomotion control (as opposed to just a reference speed), while also improving the locomotion stability by adding posture tasks in addition to the locomotion task. This control scheme is described in Appendix B.

2.2.2 Biologically inspired control

Since humanoid robots have the goal of mimicking humans in both form and functionality, it is natural that people look at them for inspiration in terms of control strategy. Although they are not restricted to their own category in a definitive way, biological inspired locomotion control approaches, specifically ones that do not explicitly use the whole dynamic and kinematic models of the robotic system, can be defined as using a more abstract model of the locomotion system, or some part of it, that is inspired by what is observed in humans or other animals.

Models that look to identify and explain the principles behind movement generation in animals are based on neurophysiologic principles, and, in some animals (including humans), describe this system as a distributed signal generation and processing, where the brain performs high-level movement control supported by a feedback system of sensors such as pressure, force, and intramuscular ones (Katz, 1996; Marder et al., 2005; Kiehn, 2006; Orlovsky, Deliagina, and Grillner, 1999).

Central Pattern Generators

The movement generation and control in humans is enacted by neural networks of the central nervous system. These networks transmit neuromuscular excitatory signals that travel in the body through action potential propagation in cells (Hodgkin and Huxley, 1952). They can produce rhythmic pattern outputs when only receiving simple input commands, without the need of sensory feedback, and are referred to as Central Pattern Generators (CPGs) (Marder and Bucher, 2001). These findings resulted from earlier studies (Jones, Tansey, and Stuart, 2011) that replaced the view that locomotion behavior resulted from consecutive muscle sensor reflexes chained together (Clower, 1998). It is believed there is a CPG unit for each limb, which are in turn composed by smaller circuits that control one muscle group of extensors and flexors of a limb (Grillner, 2011).

Although sensory feedback is not required for CPGs to produce the motion patterns, it can be used to adapt movements dynamically to changes in environment (Grillner, 2006). Signals from low-level sensors can be combined with the high-level brain signals that activate the CPGs in order to achieve different motor outputs by selecting different rhythmic patterns, and modulating the amplitude and frequency of burst signals (Rossignol, Dubuc, and Gossard, 2006), with increased stimulus resulting in higher frequency of the network rhythm. This modulation is fundamental for keeping coordination between body movements, since changing the phase of the signals changes the timing with each movement operates — the unit CPGs must be coordinated in different ways in order to generate a different activation pattern. It also can change the duration of step phases, their structure, and the transition between them (Rossignol, Dubuc, and Gossard, 2006).

Very frequently the control based on CPGs will present a rhythm generation layer, which serves as a temporal reference for a pattern generation layer. Such is the case with the implementation from Matos (2013),

$$\dot{\boldsymbol{\phi}}_i = \boldsymbol{\omega} + \boldsymbol{k} \sin(\boldsymbol{\phi}_i - \boldsymbol{\phi}_o + \pi). \tag{2.2}$$

This results in $\boldsymbol{\phi}_i$ being an increasing periodic signal that is used as the phase of the leg $\boldsymbol{i}$, with rate $\boldsymbol{\omega}$. $\boldsymbol{\phi}_o$ is the phase of leg $\boldsymbol{o}$, kept in a desired relationship with the oscillator for $\boldsymbol{i}$. The coupling strength can be controlled with $\boldsymbol{k}$. The pattern generation uses the periodic signal to control motion patterns encoded as a set of non-linear dynamical equations with well-defined attractor dynamics, which can be smoothly regulated in regard to their amplitudes, frequencies, and pattern offsets,

$$\dot{\boldsymbol{z}}_{j,i} = \boldsymbol{\alpha}(O_{j,i} - \boldsymbol{z}_{j,i}) + \sum f(\boldsymbol{z}_{j,i}, \boldsymbol{\phi}_i, \dot{\boldsymbol{\phi}}_i). \tag{2.3}$$

The position $\boldsymbol{z}_{j,i}$ of joint $\boldsymbol{j}$ from leg $\boldsymbol{i}$ is generated according to the current phase of the leg, $\boldsymbol{\phi}_i$. The offset attractor $O_{j,i}$ is a position the equations converges to if $\boldsymbol{\alpha} > 0$. Here, each function f defines a motion primitive, whose sum is used as the final output trajectory for the robot's joints.

Control based on CPGs

In his review, Ijspeert (2008) highlights the notion that CPGs usually have a significant problem with regards to parameters tuning. He notes that one of the problems to overcome in their design is the learning and optimization used to fit the sometimes dozens of parameters that shape the outputs of nonlinear dynamical systems to the desired waveforms. The waveforms themselves also have to be decided upon, and the way they translate into joint trajectories, and then to the robot's locomotion, is also not always apparent.

CPG models are frequently used in the control of biped locomotion. Most of these implementations involve sets of coupled nonlinear oscillators implemented with coupled differential equations that produce stable limit cycles — isolated periodic solutions, that are, in this case, attractive towards neighboring solutions. The phase of the oscillators can be used to control rhythmic nominal trajectories and to achieve interlimb coordination (Ijspeert, 2008). Taga, Yamaguchi, and Shimizu (1991) used coupled oscillators to model a neural rhythm generator, given the oscillatory dynamics of both the generator and the musculo-skeletal system the authors used. The coupling also allows for entrainment between those two systems, meaning they have the same period of operation. This framework, as many CPG based approaches which will be discussed here, has many open parameters, such as a nonspecific one that controls gait patterns, ones controlling feedback to the system, and also parameters related to the interconnection between unit systems. These last ones were set manually in a way that modulates amplitudes and relative frequencies to produce specific joint motions, which may or may not be optimal. It is not clear whether the parameters related to feedback were set manually or optimized in some kind of way.

A later revision to Taga's control scheme came with the objective of making the model of the musculo-skeletal system step over visible obstacles (Taga, 1998). The rhythm generator was combined with a discrete movement generator that, receiving visual information regarding the obstacle, modifies the gait pattern. This generator uses modification signals whose amplitudes are controlled by different parameters, and have to be adjusted to produce smooth and coordinated changes in the gait. The author also noted that choosing appropriate values of these parameters and ones that control step length is important for obstacle clearing, but does not clarify how that choice is made. Additionally, there are parameters that determine the strength of the torque. In short, both the original control framework and its revision have open parameters that affect the behavior of the locomotion task and do not have a straightforward way to be tuned toward objective optimization or even basic, stable locomotion.

A few years later, Aoi and Tsuchiya (2005) developed a system using nonlinear oscillators to setup the rhythm of a trajectory generator. Their phases can be reset, and nominal trajectories can be modified by sensory feedback related to robot posture and motion. The ratio between the durations of the stance phase and the step cycle, angular velocities, and trajectory parameters need to be set. These parameters are tuned depending on fixed environmental situations and choice of nominal speed, in a manual way.

Gen Endo et al. (2005) also developed a control system based on a CPG model, this one consisting of neural oscillators that use posture and feet reaction forces feedback to stabilize the locomotion. The parameters of these oscillators were manually tuned, and feedback related coefficients where obtained empirically through "numerical simulations and hardware implementation". In the paper, the authors mentioned wanting to address these optimization issues with a learning framework. In a related study, Morimoto et al. (2006) used coupled oscillators to adjust the phase of nominal joint trajectories. The phase of inverted pendulum dynamics used for stepping and walking is adjusted according to the position and velocity of the CoP, and this adjustment functions as modulation of joint trajectories. The system uses an objective function to find a linear feedback controller that maximizes its components, but the functions itself has parameters that need to be set. Additionally, joint trajectories are modulated by amplitude parameters, and phases, frequencies, and coupling constants also need to be tuned.

In Aoi and Tsuchiya (2011) the authors conducted a local stability analysis of a locomotion control system in a study that also encompasses parameter tuning. The system uses nonlinear oscillators with a stable limit cycle to generate joint motions. Phase-resetting using foot-contact information modulates the oscillator's phase, which, in turn, modulates its amplitude. A gain parameter controls the strength of these interactions that results in stable movement, and is therefore an important one to tune to achieve smooth locomotion. They analyzed the effect this parameter had in stability through numerical analysis to determine the values that lead to optimal stability. They also investigated the effects of feedback parameters and observed that for a large range the joint trajectory errors and the ratio between horizontal and vertical ground reactions forces remained small. The numerical analysis is specific to this system and robot model tested, and it makes assumptions about their dynamics. Furthermore, these optimizations show that there are indeed ranges of parameters that show better results and stability, hence reinforcing the notion that their optimization is of interest.

Motion Primitives

Ijspeert et al. (2013) presented a review on modeling the attractor behaviors of nonlinear dynamical systems by using statistical learning techniques. They developed a model of learnable nonlinear attractor systems that is able to encode both rhythmic and discrete movements by using the concept of motor primitives, which consist of a modular approach to movement generation where it results from the combination of different stable motions. They call this system dynamic movement primitives (DMPs), because they see these attractor systems as "building blocks that can be used and modulated in real time for generating complex movements". The DMPs can encode, for example, trajectories that are directly fed to each joint, or kinematic trajectories that are transformed with inverse kinematics to joint trajectories (Kuppuswamy and Alessandro, 2011; Hiratsuka et al., 2016). The authors highlighted how their proposed system is a more general approach to attractor systems modeling, since it uses the same form of representation for different applications, while relying on machine learning to

adjust the model to observed data. They also note the design of spatial coupling terms is a "research topic by itself" and the choice for these variables are "critical" and often need to be altered according to different objectives. Furthermore, they point out that optimization approaches can be used to tune the open parameters of the dynamic systems model, despite their approach being flexible and applicable to a broad range of control problems.

As an example of the optimization of the open parameters of these types of systems, Schaal, Mohajerian, and Ijspeert (2007) combined the self-organizational approach of systems based on nonlinear differential equations with the principle of optimizing a control task to general principles such as minimizing trajectory errors or maximizing speed. Part of the work deals with learning and optimization with DMPs, where they used supervised and reinforcement learning (discussed in Section 2.2.3) to find the weights of a function used to shape the desired trajectory of the nonlinear dynamic system. The learning approaches involved shaping the trajectory to a desired one, which requires a way to obtain a suitable trajectory for every situation to be used as an adaptation tool. Parameter optimization was done by minimizing some of the terms, or combinations of terms, of the DMP equations. This effectively allows to shape the trajectory to some desired optima (like minimum torque change), as long as that feature is represented in the DMP itself, which means optimizing for features not included in the equations is not possible with this approach.

Matsubara, Hyon, and Morimoto (2011) introduced a framework where DMPs are encoded as parametric trajectories which can be learned from multiple demonstrations. Common factors from these demonstrations are extracted and shape the trajectories, which parametric nature allows for scalability and generalization towards different applications and environments. The scalability is encoded in a "style" parameter that represents different motions (*e.g.* different walking styles), and many different styles can be gathered from the learning process, with intermediate values resulting in interpolated functions. This adds a way to synthesize the open parameters and allow for faster and easier optimization by tuning the style parameter, but also reduces the flexibility that multiple parameters would provide.

Hauser et al. (2011) approach the problem of linear balance control in a bipedal robot by using movements generated through linear superposition of a few "stereotypical combinations of simultaneous movements of many joints", referred to as kinematic synergies (KS). They based this concept on observations done on biological organisms, proposing they use these synergies to reduce the high dimensionality of their action space when walking. They postulate that "no special tuning of the controller parameters" is required in order to maintain the robot balance, since this occurs for a wide range of these variables. One KS is controlled by only one parameter and affects multiple joints, therefore reducing the control space. The KSs are obtained offline using calculations based on the kinematic model and masses of the robot, and fixed during control. The control system was tested for multiple combinations of feedback control parameters and different values of the length and mass of the robot. Although the results show low tracking errors for these combinations, they also show that some of

them are indeed better at reducing error, and others at achieving a greater range of length and mass with which they can successfully track the desired joint trajectories. This variability and trade-off shown by the results give credence to the notion that some kind of optimization for the process is desirable.

With coupled dynamic oscillators a lot of their parameters are chosen in a analytic fashion, in order to achieve limit cycles or coordination through phase manipulation. It is not always clear if these characteristics are optimal for the locomotion process, and, in most cases, there are parameters that are chosen manually, or with a basic linear search, which may ignore controllers that produce potentially faster or more energy efficient behaviors. Choosing a specific set of joint trajectories (Taga, 1998) is usually based on broad leg and body movements during the locomotion, without great specificity or evidence of these producing optimal behaviors. In some cases parameters are chosen specifically to adapt to certain environmental characteristics, or locomotion features such as nominal speed (Aoi and Tsuchiya, 2005), but in a case by case basis and without an automated and consistent method.

2.2.3 Biped locomotion controller optimization

The previous sections explored some issues related to the stability and efficiency towards certain objectives present in multiple locomotion control approaches in general, and specifically biped locomotion frameworks. We also noted that differences in behavior that can bring greater stability, speed, or energy efficiency, for example, may be achieved by tuning some of the parameters of the control system.

Although these issues were mentioned or looked into to be solved in some of the works discussed in the previous sections, they were not their main focus. This section explores approaches that focus on optimization of biped locomotion controllers towards different goals.

Analytical supported optimization

One way to approach controller optimization is by supporting it with analytical studies of the dynamic of the robot's locomotion. In specific cases, where the complexity of the interactions between the systems in relation to the desired process to be optimized is low enough, this approach may be the most suitable in terms of time expenditure.

As an example of an analytical approach, Panne, Fiume, and Vranesic (1992) presented a controller for the locomotion of a planar model of a biped robot that models its dynamics, the current state of the robot, and information from the environment to calculate a vector of inputs to the robot's joints. The solutions obtained from this analysis are in the form of a lookup table with the control solution corresponding to a specific space-state of the task.

In a more recent example, Wang et al. (2012) presented a 3D model of a humanoid robot that contains muscolotendon models in each leg, with biologically motivated control laws. The parameters of the controllers are set by minimizing a biological

model of metabolic energy expenditure while satisfying a number of locomotion task terms.

Zheng and Yamane (2011) worked on a analytic approach to the optimization and control of a biped walking on a rolling cylinder. The biped is simulated as a simplified linear model, and a collision model was derived, consisting of the cylinder and two legs. With a balance controller coupled with the collision model, the motion of the biped is uniquely determined by the initial state of the system. The robot can be made to achieve a cyclic gait by optimizing the initial state with a cost function with a constraint on the desired average rolling velocity.

In other situations, the control framework has parameters that control aspects of the locomotion behavior in an intuitive and direct manner. Lee and Ryoo (2014) proposed a tuning system for a ZMP based control scheme that takes advantage of the fact that quantities like feet positions offsets and the step period can be directly controlled.

Analytical approaches conclusions

Although applicable in the right conditions, analytical studies of models of the robots and their environments can be ineffective because of the complexity of these systems and the interaction between them, as pointed out by Slatton et al. (2008). They are not good as universal approaches, due to the fact that they usually are controller specific or even robot specific, or need a significant amount of work to adapt to different models. In a survey of the modeling and control of bipedal robots, Hurmuzlu, Génot, and Brogliato (2001) postulated that a concise and sufficiently general theoretical analysis framework that allows one to take into account hybrid dynamics when designing controllers was still missing.

Derivative-free optimization

When an analytic approach is impractical, an empirical trial and error approach might be effective. The information needed for the optimization of the controllers' parameters may be gathered by experimentation that provides information for derivative-free optimization methods. These need dozens, or sometimes thousands of evaluations to reach acceptable solutions, which is generally only feasible by using a simulation of the robot's locomotion that can be sped up relative to real time, and possibly ran in parallel to other simulations.

Evolutionary algorithms

Some examples of these methods employed in robotics control optimization are evolutionary algorithms (EAs), which are inspired by biological evolution. Examples of EA approaches are genetic algorithms (GAs), the covariance matrix adaptation evolution strategy (CMA-ES), and particle swarm optimization (PSO). GAs (Mitchel, 1998) are optimization methods based on a search heuristic that mimics the process of natural selection. They use a set of candidate solutions that is evolved through several generations. The CMA-ES method (Hansen, 2006), also part of evolutionary computation,

updates the covariance matrix that represents dependencies between the variables of the new candidate solutions of the evolution. PSO (Kennedy and Eberhart, 1995) also uses a population of candidate solutions (a swarm of particles), which in this case is moved around a search space towards better solutions according to the particle's position and velocity.

Rodrigues et al. (1996) used a GA to obtain a desired trajectory for a biped's CoM. They find the necessary torques of each joint of the model by optimizing input commands to the joints directly. This is an example of an approach that uses optimization intrinsically in its control, since it outputs the torque trajectories, instead of using this optimization to change the controller's behavior by tuning its parameters.

Wu and Popović (2010) described a framework for the optimization of a biped locomotion controller that uses a model based approach with dynamics informed step planning, similar to the approaches described in Section 2.2.1. The parameters of the footstep planner are tuned in an offline optimization towards different goal trajectories, using a black-box CMA-ES evolution strategy. This approach is controller specific, and needs different optimizations for each trajectory. The controller includes online adaptation to slope and height changes, but the information about these obstacles is directly fed back, without there being any kind of sensor information involved.

Working with dynamical systems encoding, Kieboom (2009) proposed a method to design a locomotion framework independently of the type of robot. Part of the methods involves using PSO algorithms to evolve a gait encoded into a CPG network, using the mean walked distance as the fitness function. The author attributes part of the success of the method to user top-down decisions on parameter constraints that minimize the parameter space and make this type of optimization algorithms faster and more reliable.

Paul and Bongard (2001) approached the controller optimization of a biped robot problem from a mechanical design standpoint, by changing the mass distribution of a physics simulated 5-link biped robot along with the evolution of the neural controller using a GA. This approach of combining mechanical design with controller optimization proved to be useful in this particular work, where the overall optimization results were better than those of a GA evolution of the controller parameters alone.

Bayesian optimization

Another derivative-free approach is Bayesian optimization, which is applied to black-box functions by modeling it as a group of random distributions and using priors that incorporate the information gathered from past data to make assumptions for future evaluations. The model selects points of the function to sample, which are then used to update the posterior distribution of the function (Mockus, 1989).

Calandra et al. (2016) evaluated Bayesian optimization approaches to automatic gait optimization. They highlighted the efficiency of the method in using past interactions, and the subsequently fewer number of interactions with the model or real robot needed to reach desirable behavior.

Working with this idea that a small number of experiments is a requirement when

working with a real robot, Hemker et al. (2009) proposed an optimization scheme for the walking speed of a humanoid robot that uses a surrogate function of the locomotion process. The surrogate is a way to measure the outcome of the locomotion by measuring an outcome of its model instead. The surrogate is started with a prior given by a slow but stable walking motion, and is then sequentially updated. In these updates, points are sampled by varying the parameters' values by a certain percentage, and then evaluating the maximum value of the surrogate function after taking those samples into account. The process is stopped after a desired threshold is attained. The optimization was applied to the parameters of a trajectory generator with low dimensional inputs.

Reinforcement learning

Reinforcement learning (RL) is a trial-and-error type of machine learning where an agent's behavior is optimized by making it act in a way that maximizes its rewards. In biped locomotion control, the agent, being the control framework, can, for instance, try to create a controller that maximizes an expected total return, expressed in the form of a function. The RL algorithm defines the behavior of the control framework with a policy. Although good for non-linear optimizations, RL approaches are difficult to apply to robot control in general, due to the usually high-dimensional action spaces and continuous state natures of the problem, and, consequently, the high number of trials required to reach a good policy (Kober and Peters, 2012).

Fujiki et al. (2012) used RL to improve the stability of a locomotion control system of a biped walking in a split belt treadmill. The system consists of nonlinear oscillators with phase resetting, and they use RL with the relative phase between the leg oscillators as a cost function they look to minimize. Controlling the interlimb coordination is important for the feasibility of the motion, but this approach is not geared towards optimizing performance features.

Authors from a study (Morimoto et al., 2007) mentioned in Section 2.2.2 proposed an improvement on the control system that uses locomotion dynamics approximated with RL. A Gaussian process (GP) is used to approximate the dynamics of the locomotion. A GP is a statistical model used to estimate a probability distribution for a target function, and is defined by given mean and covariance functions. RL was used to affect the GP model towards better performance in the tasks of stepping and walking, given that the approximated dynamics modulate the amplitude of sinusoidal patterns produced by dynamic oscillators. Modulating a relationship between approximated dynamics (here using an inverted pendulum model) and sinusoidal patterns can work, if the approximation is accurate enough. It, however, causes a loss of flexibility by restricting some of the control parameters to that relationship, and leaving others to be tuned manually. Additionally, a reward function that contains only one value doesn't provide choice between controllers optimized towards different sets of objectives.

Derivative-free approaches conclusions

Derivative-free approaches are more broadly applicable than analytical approaches to biped locomotion control, although they may lose effectiveness for more specific applications, as well as require dozens or thousands of trials for data collection. They can be used in any control framework that is parameterized in components that affect the locomotion behavior, and can be setup in a more or less universal way if the performance indicators they use are appropriate for multiple humanoids and situations, which is feasible when working with a specific type of robot (bipedal), and a specific type of task (locomotion). Some methods, such as RL, require specific representations of the parameters tuned and the performance(s) evaluated (such as policies and rewards), that may need to vary for different control approaches. This blurs the line between some derivative-free approaches and analytical ones, which in general represent a trade-off between universality of application and efficient in time (given that the analytical approach is not too complex) and effectiveness in performance (if the derivative-free approach does not achieve an optimal result) (Michalewicz and Fogel, 2004).

2.2.4 Biped locomotion controller adaptation

Besides tuning a humanoid locomotion controller towards optimizing certain objectives, adaptation to different terrains (or other changes outside the robotic system) is also desirable. Some control frameworks specifically tackle this problem, usually implementing ways to adapt to specific terrain changes (*e.g.* different slopes).

As with optimization approaches in the previous section, some adaptation approaches were touched on throughout this chapter, but in this one we give them greater focus.

Adaptation to slope changes

Like the optimization towards different objectives, solutions for the adaptation required by terrain changes vary in their complexity and approach. As an early example of a simple way to adapt to an often tackled terrain change, Panne, Fiume, and Vranesic (1992) introduced adaptation to arbitrary slope values by using linear interpolation between control tables to generate a control solution. This approach is dependent on a linear relationship between changes in both the control solution and the slope the robot is intended to walk on, or at least one close enough to still provide stable locomotion.

Using a more advanced approach to the slope changes issue, Liu, Wang, and Chen (2013) proposed a CPG based control system that outputs a CoM trajectory that can then be mapped from the workspace to a joint space. The system is setup in a way that the biped's step length, step height, and the duration periods of the swinging and supporting phases of the legs can be adjusted in real time. They showcase this adaptation capability to adapt to changes on the environment by having the robot move in sloped terrain and using its body attitude to determine its step length, and maintain stability. While this approach works, it is specifically designed to adapt

to slope changes, and a more general approach can make more extensive use of the controller's ability to change these locomotion features.

Adaptation to uneven terrain

Some approaches focus on terrain with sudden height changes, such as steps, as opposed to changes in slopes. Hopkins, Hong, and Leonessa (2014) developed a framework for humanoid locomotion adaptable to uneven terrain. Their control based on a linear transformation of the CoM state allows for variable CoM trajectories during stepping on terrain with variable height. The control system uses a nominal CoM trajectory, and its dynamics, to find admissible trajectories given desired references for the ZMP, and assumes the desired step positions are provided by a high level footstep planner. This approach includes the height changes in the CoM trajectories, rather than using feedback to obtain the information, making it not applicable to situations where the changes are not known beforehand.

In the context of using vision to aid legged locomotion on difficult terrain, Gay (2014) applied a mapping between sensory information and modifications to the dynamics of a CPG controller, which were learned using a neural network. They applied this approach to a biped balancing in a moving platform and quadruped walking in uneven terrain. The main downside to this approach is it being specific to this CPG controller because of how the feedback is mapped to the changes in the controller.

Adaptation applicable to different terrain changes

Kimura, Fukuoka, and Cohen (2007) approached the problem of walking in irregular terrain using a more complex and broad strategy, looking to adapt to different types of terrain changes. They designed the control system to comply with expected necessary conditions for uneven terrain. It consists of a CPG system with responses and reflexes related to muscles that are modeled as PD-controllers at the robot's joints. They proposed the necessary conditions for stable dynamic walking on irregular terrain, and designed the responses and reflexes of the control system to comply with these conditions. They successfully tested the control system both in an indoor and in an outdoor, natural environment. This approach is dependent on a careful design of the robot, and the reflexes system entrainment with the CPG. The fact that it is applied to a quadruped, in itself, implies the need for great changes in order to use it in a bidepal robot.

In a similarly broad type of adaptation approach, Wang et al. (2010) developed a method that optimizes locomotion controllers for robustness to a general class of uncertainties. The method represents variables such as external forces and control torques as probability distributions, which, together with a controller and the locomotion's dynamics, define the probability distribution of single motions. A return function is then used to score the quality of a given motion, which is used to optimize a controller by maximizing its expected return. The flexibility of the adaptation scheme was demonstrated by applying it to different situations, such as external disturbances and a

slippery surface. The authors noted that some adaptations appear unnatural, and their method was only applied to a simple four state PD controller — optimization for more complex ones may become more difficult, since they use dynamics of the locomotion for the expected returns.

To conclude, we highlight a framework that applies a broad approach to both the problems of optimization and adaptation to changes in the locomotion process. The multi-dimensional archive of phenotypic elites (MAP-Elites) (Mouret and Clune, 2015) is a search algorithm that produces a map which is by design diverse on the feature space. An intelligent trial and error algorithm (Cully et al., 2015) that uses the information from MAP-Elites is then used to conduct experiments with the objective of discovering a compensatory behavior that works in spite of the damage to a robot. It optimizes the control parameters without sensory feedback, or feedback from quantities internal to the controller, making this approach more general than the previous ones, allowing it to be implemented in different controllers, robots, and contexts without being to be fundamentally changed. The drawback is the loss of adaptability to these very specific situations that other works are focused on.

2.2.5 Conclusions

This section provided a literature review focused on works that cover the aspects of humanoid locomotion control identified in Chapter 1 as central to this work: 1) Controllers have open parameters that can be optimized towards different goals; 2) There are multiple optimization approaches, with different advantages and disadvantages; 3) These parameters can also be tuned to adapt to different environmental changes.

The reviewed papers were discussed with the above points in mind, and also taking into account the fact this thesis is looking for an approach that is universal in terms of its application towards different controllers, objectives, and environment changes.

The possibility and need for controller optimization

The model predictive control scheme introduced by Kajita et al. (2003), and further developed by Wieber (2006), Diedam et al. (2008), and Herdt, Perrin, and Wieber (2010), produces a parameterized locomotion trajectory, defined by quantities such as the length or duration of each step. Changing the precision of the preview dynamics, or the optimization of the state vector (*e.g.* the joints' accelerations), can change the outcome of the locomotion, but usually in a less impactful way than changes to the trajectory parameters. The task based approach introduced by Salini, Padois, and Bidaud (2011) further expands the possibilities for optimization, since it adds the possibility to give more or less weight to the CoM trajectory from the predictive control, or a posture task, for example.

Rhythm generators, usually based on biological inspired approaches, tend to have a different relationship between their parameters and the locomotion behavior. Instead of directly changing the height or width of a step, modulation of CPG signals change the way different patterns combine to produce joints' outputs, or a locomotion trajectory

(Rossignol, Dubuc, and Gossard, 2006). There are multiple parameters that affect multiple patterns, and the behavior produced by their combination can be hard to predict. These factors make the optimization of these parameters complicated, as noted by Ijspeert (2008).

Taga (1998) implemented a visual feedback system in a rhythm generator, but parameters affecting the feedback signals, and the steps length, have to be tuned. Aoi and Tsuchiya (2005) presented a rhythmic trajectory generator that contains (manually tuned) parameters that control the duration of the stance and swing phases of locomotion, as well as angular velocities. The same is true for Gen Endo et al. (2005), who manually tuned some of the parameters of their CPG, mentioning their intention to address the optimization issues with a learning framework.

In Aoi and Tsuchiya (2011), a study on the tuning of feedback parameters of a system of nonlinear oscillators showed that, for some of the ranges tested, the observed locomotion was more stable. Hauser et al. (2011) used a system with superimposition of different joint movements, which parameters were tuned by testing for multiple combinations of feedback control parameters and the length and mass of the robot. The results show that some of these combinations are better for reducing the trajectory error, while some values of the feedback parameters are better at successful tracking trajectories for a greater range of length and mass values.

This section of the review covered very distinct control approaches, which, to a lesser or greater degree, all show the same general point: differences in locomotion behavior, which can result in greater stability, speed, or energy efficiency, for example, may be achieved by tuning some of the parameters of the control system.

Optimization approaches

After establishing the need for the optimization of these controllers, the review focused on some of the usual approaches to this problematic.

Analytical studies of the system's dynamics can be ineffective because of their complexity (Slatton et al., 2008). They usually are controller, or robot, specific, or need a significant amount of work to adapt to different models.

Derivative-free optimization can provide an alternative approach that does not need a mathematical model to be applied. It can be used in more general ways (i.e., not controller/robot specific), but usually need more time to obtain good results. These usually can be used in any control framework that contains open parameters. Some of these approaches, such as reinforcement learning, require specific representations of variables optimized and objectives optimized for, which may vary for different control approaches.

Adaptation approaches

Besides optimizations towards different locomotion features, these controllers can also be adapted to suit environment changes, such as changes to the terrain or the robot model.

A common terrain change is encountering different slopes. Panne, Fiume, and Vranesic (1992) introduced adaptation to arbitrary slope values by using a linear interpolation approach that assumes a linear relationship between changes in both the control solution and the slope the robot is intended to walk on. Liu, Wang, and Chen (2013) proposed a CPG based control system that can change values like the length or height of a step in real time, using body attitude as feedback to adapt to slope changes. These approach work for this specific problem, but are not general towards other adaptations.

Another common problem appears with uneven terrains (i.e., showing variable height). These tend to have the same problem with lack of flexibility. Hopkins, Hong, and Leonessa (2014) developed a control system that uses nominal CoM trajectories to find admissible steps given desired references for the zero moment point. The controller adapts to height changes in the terrain by incorporating them in the predefined CoM trajectory. Gay (2014) used sensory information to modify the dynamics of a CPG controller. This can be applied to adaptation to walking in uneven, but is controller specific, and also objective/adaptation specific because these are dependent on the sensors used.

Using a more general approach, Cully et al. (2015) create a map of solutions diverse towards different locomotion features, and then use it to adapt to terrain or robot changes during the locomotion process.

Overview of each paper's features

Table 2.1 shows an overview of the papers reviewed on sections 2.2.3 and 2.2.4, taking into account which of the required features highlighted on Chapter 1 they present.

The approach from Cully et al. (2015) is the one that gets closer to the design requirements introduced previously. This approach can be used with any controller with open parameters, and adapt to different environment changes. It does, however, have some problems in this context.

The map of solutions is dependent on defining features that may need to be specific to the robot beforehand (*e.g.* duty factor of each leg in a hexapod), and obtaining a diverse map may not be ideal, since it intentionally spends time obtaining possibly low performing solutions. In the adaptation process, the environment changes are not encoded in variables, which makes it more flexible, but also requires the Bayesian optimization to re-learn the relation between features and performance. This relation may be too complicated to represent with Gaussian processes, or to much time/computational expensive.

Table 2.1: Summary of the works reviewed in this chapter, highlighting relevant features. "Controller optimization" provides a small description of the optimization approach, "objectives" lists the optimization objectives, "adaptation" the types of adaptation the framework allows for, and "controller independent" lists whether the approach is applicable to any controller with open parameters. If the approach is applicable to any optimization objective, or adaptable to any terrain/robot change that can be measured during locomotion, the corresponding column will list "any". Cells where a feature is absent are marked as "-".

Paper reference	Controller optimization	Objectives	Adaptation	Controller independent
(Panne, Fiume, and Vranesic, 1992)	Linear interpolation	-	Slope	-
(Wang et al., 2012)	Biological model energy minimization	Metabolic energy expenditure	-	-
(Zheng and Yamane, 2011)	Collision and balance models	Speed	-	-
(Lee and Ryoo, 2014)	ZMP parameters manual tuning	Any	-	-
(Rodrigues et al., 1996)	GA applied to joint torques	Any	-	Yes
(Wu and Popović, 2010)	CMA-ES applied to step planner	Trajectory	Slope and uneven terrain	-
(Kieboom, 2009)	PSO applied to CPG network	Any	External forces	Yes
(Paul and Bongard, 2001)	Changes to robot's mass distribution	Any	-	-
(Hemker et al., 2009)	Bayesian optimization of trajectory generator	Any	-	-
(Fujiki et al., 2012)	RL applied to nonlinear oscillators	Interlimb coordination	Walking on splitbelt treadmill	-
(Morimoto et al., 2007)	RL applied to a GP model of dynamics	Speed	-	-
(Liu, Wang, and Chen, 2013)	Automatic adaptation of steps characteristics	-	Slope	-
(Hopkins, Hong, and Leonessa, 2014)	Automatic adaptation of CoM trajectory	-	Uneven terrain	-
(Gay, 2014)	Sensory information affecting CPG dynamics	-	Moving platform	-
(Kimura, Fukuoka, and Cohen, 2007)	Design of CPG with modeled muscles	-	Uneven terrain	-
(Wang et al., 2010)	Probably distributions of different forces	-	Any	-
(Cully et al., 2015)	Bayesian optimization of solution-features relations	Any	Any	Yes

Chapter 3

Background

Optimizing the locomotion control of a robot can be done by treating the system as something that converts numerical inputs (the parameters of the controller, and context variables) into numerical outputs (the features of locomotion). Mathematical optimization can then be applied to that system, resulting in finding the solutions (sets of parameters) that allow the robot to walk with the best performance possible.

This chapter introduces the concept of mathematical optimization, and focuses on metaheuristics — procedures that provide good solutions in the absence of a complete representation of the system, or in the case of limited computation capacity. The topic of optimizing towards multiple objectives is also covered, since multiple locomotion features are considered in this work. These features are used as optimization objectives.

One specific optimization algorithm was used throughout the entire work - the non-dominated sorting genetic algorithm II (NSGA-II). A detailed description of this procedure is included along with the concepts of evolutionary algorithms, which NSGA-II is a part of, and Pareto efficiency, which is important for multi-objective optimization.

3.1 Mathematical optimization

When presented with a system that can be controlled through variable quantities, one can select the inputs $\boldsymbol{x}$ that produced a given output $f(\boldsymbol{x})$, with f being the function that represents the relationship between the input and output of the system.

The function f may represent a cost or performance function that we want to optimize, either by maximizing or minimizing its output, which is done by changing its inputs. These inputs, in the context of this work, will be referred to as controller parameters. This process looks for $\boldsymbol{x}^*$, which is the set of parameters that either maximizes ($f(\boldsymbol{x}^*) \geq f(\boldsymbol{x})$, for all possible $\boldsymbol{x}$), or minimizes ($f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$, for all possible $\boldsymbol{x}$) the output.

If there is an analytical representation of f, the output can be calculated. Furthermore, the derivative of the function, if it exists, can also be calculated, which gives information related to how the output of the system is affected by small changes to the parameters. If one can determine when the derivative is null, this information can be

used to find optima (maximum or minimum values) of the function. It cannot be used, however, to distinguish between local and global optima.

In some situations, an exact derivative of the function is hard or impossible to obtain. When there is no analytical representation of f available, or it is hard to calculate, the outputs of the system can only be observed, and not calculated [1]. An alternative in this situation would be to manually compute the derivative of the function, and use it to find the optima. This is the approach used in the gradient descent (to find a minimum), and gradient ascent (to find a maximum) methods (Snyman, 2005). A problem with this approach is that measures of the output are discontinuous, and are often affected by noise, which makes it more difficult to use (Amaran et al., 2016). These issues mean that the optimization process cannot be conducted in the straightforward way of using the information provided by the mathematical representation of the function and its derivatives.

3.1.1 Metaheuristics

When finding an optimal value is not possible, due to lack of an exact derivative or other constraints (*e.g.*, high computational demand), an alternative approach is to use metaheuristics. These are procedures that do not find exact solutions, but instead look for ones that are "good", albeit not optimal. They use few observations about the inputs and respective outputs of the function to make assumptions about the whole system.

A heuristic is an approach to problem solving that finds approximate solutions in a faster way than exact methods. A metaheuristic is a procedure that is used to select a heuristic that may lead to a good solution in an optimization problem, *i.e.* it guides the search process (Deb, 2001). Metaheuristics are usually faster than an algorithm providing an exact solution (Adekanmbi and Green, 2015; Rios and Sahinidis, 2013), but they do not guarantee a convergence towards the global optimum. Many of the algorithms in this category use stochastic optimization, which generates and uses random variables, meaning randomness may affect the optimization process.

Exact methods usually require specific formulations of the optimization problem, such as objective functions expressed as linear functions of key variables. Due to the fact they are defined in general terms, metaheuristic algorithms can be adapted to optimization problems which are allowed enough computation time, and can offer a good solution reasonably fast (Talbi, 2009).

Classes and examples of metaheuristics algorithms

The goal of metaheuristic algorithms is to find the best possible solutions to an optimization problem. They use an encoding of a solution that can be stored in a predictable way, and manipulated with different mathematical operators. The general process in these optimizations is to evaluate potential solutions, and change them with

[1] As an example, a humanoid robot walks with a speed that can be measured, but not always calculated beforehand with high precision.

different operations, in order to find different solutions (Glover and Kochenberger, 2003).

There are many different metaheuristics algorithms, varying in their approach and applicability. Three classes can be distinguished, differentiated by the way they manipulate solutions: local search metaheuristics, that make changes to a single solution through multiple iterations; constructive metaheuristics, which iteratively construct solutions from constituting elements (*e.g.* a real number at a time); population based metaheuristics, which work with multiple solutions simultaneously, and combine them into new ones. These classes are not the only way to distinguish between the various methods in this field, and these algorithms can combine approaches from each class (Glover and Kochenberger, 2003).

Local search metaheuristics

When solving computationally expensive optimization problems, local search can be a preferred approach for the task. This is a metaheuristic approach to the gradient descent algorithm, where the search will move from solution to solution in a space of candidate solutions that is obtained by applying small changes to them. Since it only needs to store information related to the small area being explored, local search can lessen the negative impact in memory and time of computationally intensive problems (Glover and Kochenberger, 2003).

An example of such an algorithm is the hill climbing algorithm (Russell, Norvig, and Davis, 2010), which starts with a random solution, and incrementally makes changes to it until a better one cannot be found. A single element of the solution is changed at a time, and any solution that produces a better result is kept.

A more complex approach, called simulated annealing (SA) (Khachaturyan et al., 1981; Kirkpatrick, Gelatt, and Vecchi, 1983), mimics the annealing process of a crystalline solid in its movements to find different solutions. In each iteration, a random solution is selected from the neighborhood of the current one. This neighborhood is defined as the set of solutions that can be reached by making a single move to the current one. A probability of being selected is attributed to the selected solution, which depends on the performance of the solution, but also on the temperature of the process, which cools down with passing iterations, making it less likely for the solution to change significantly, or at all.

Constructive metaheuristics

Constructive metaheuristics have a distinctive way of obtaining new solution than both local search and population based approaches. These solutions are iteratively constructed from their parts, instead of being altered versions of former, complete, solutions. A local search after the construction phase is often used to improve the quality of the initial solution (Glover and Kochenberger, 2003).

An example of such algorithms is the ant colony optimization (ACO) technique (Dorigo, Maniezzo, and Colorni, 1996). This term refers to constructive metaheuristics

that build solutions by mimicking the paths ants follow when looking for food. When returning from successful foraging, ants return to their colony while leaving behind pheromones, which are used by other ants to find the same food source. This behavior can be mimicked by an algorithm by using a parameter called the pheromone level. The algorithm runs multiple agents (ants) in parallel, each looking to construct a solution. Once this is achieved, the pheromone level of each element of the solutions is update, with better solutions being allocated more pheromones. When the process of solution construction is repeated, elements from well performing solutions will tend to be selected more often.

Population-based metaheuristics

Population-based metaheuristics can be used when optimizing a function with several local optima, or when finding a global optimum is especially important. These algorithms find new solutions by selecting and combining solutions from the current set (called population), therefore creating a new one. They usually look for solutions over a vaster search space, making it more likely to find a global optimum (Rios and Sahinidis, 2013). Some of these techniques are based on local search approaches, although they perform a global search.

Particle swarm optimization (PSO) is a population-based metaheuristic that treats its population of candidate solutions as a swarm of particles, that are moved around the search-space of the optimization by applying formulae to the particles' positions and velocities (Kennedy and Eberhart, 1995). Evolutionary algorithms (EAs) are a vast group of metaheuristics that use mechanisms observed in the field of biological evolution to affect its population of candidate solutions.

All of these approaches have their own advantages and disadvantages, and the choice of application is largely dependent on the type of function one wants to optimize, as well as how precise and/or diverse one wants the final solution, or collection of solutions, to be (Eiben and Smith, 2003). Throughout the rest of this chapter, and also throughout the rest of this dissertation, the focus will go towards EAs, due to their more extensive use in multi-objective optimization.

3.2 Evolutionary algorithms

EAs involve techniques that are inspired by biological evolution, which main aspects are variation (through mutation and recombination), and selection (Eiben and Smith, 2003). They are a population-based type of metaheuristic optimization algorithms. EAs look for approximate solutions, and do not make assumptions about the fitness landscape, making them good candidates for optimizing complex problems that are dependent on multiple variables, such as biped locomotion (Eaton, 2015).

EAs can be separated into multiple types, differing in the way their encode solutions, and what the particular biological inspiration for that algorithm is. Genetic algorithms (GAs) are inspired in the process of natural selection, and represent solu-

tions in the form of strings of numbers. They use selection, mutation, and crossover to optimize the solutions. Evolution strategies (ES) use these same types of genetic operators, with different selection processes, and usually self-adaptive mutation rates. They encode solutions as vectors of real numbers . Other examples of solution encoding are representing the genomes as computer programs (called genetic programming), and as artificial neural networks (in an approach called neuroevolution) (Eiben and Smith, 2003).

The general steps taken in an EA (Eiben and Smith, 2003) are as follows. First, an initial population of individuals is randomly generated; this is the first generation. A fitness value is assigned to each individual solution, after evaluating them using the function to be optimized. The fitness is a value that translates how well an individual is suited to solve the optimization problem. In each generation, the individuals with the best fitness values are selected as the parents of the next generation. These are used to breed through crossover and mutation, leading to new individuals. A crossover takes multiple parent solutions and produces a child from them, while a mutation alters one or more values of a single solution in order to generate a new one. These new individuals are evaluated to find their fitness, and the least fit individuals from the population are replaced, if they show worse performance. This process is repeated for a set number of generations, or when an acceptable group of solutions is reached.

3.2.1 Multi-objective optimization

The optimization and adaptation problems tackled in the course of this document use multiple optimization objectives, with $f(\boldsymbol{x})$ being represented by multiple values, rather than a single one.

This approach is useful when optimizing humanoid locomotion, since optimizing for speed only, for instance, could cause the system to be unstable, or energy inefficient. As discussed in the previous chapter, this task is very complex and hard to optimize, and optimizing various aspects of the system simultaneously is crucial.

Optimizing a function towards multiple objectives requires a way of making decisions about the performance value (referred to as fitness) of each solution, since the output is represented by multiple values, and not a single one that can directly be compared to the fitness of another solution (Miettinen, 1998). One way to deal with this is the scalarization approach, which transforms multi-objective problems into a single-objective problem using a weighted sum of the fitness values (Jahn, 1985).

Because of the possibility of not finding a solution that simultaneously optimizes each objective, the concept of Pareto efficiency (presented in Section 3.2.2) is used in the performance evaluation of each solution. The scalarization approach reformulates the problem in a way that guarantees that the optimal solutions for the new single-objective optimization are Pareto optimal for the multi-objective optimization (Jahn, 1985).

3.2.2 Pareto efficiency

When optimizing for multiple objectives, a set of solutions is selected based on the concept of Pareto optimality (Deb, 2001; Tomoiagă et al., 2013) (named after Vilfredo Pareto). A solution is considered Pareto optimal, or nondominated, if there are no other solutions which corresponding outcomes are all better in value than, or equal to, those of said solution. In other words, a solution $\boldsymbol{x}'$ is said to dominate another solution $\boldsymbol{x}''$ if:

1. $\boldsymbol{x}'$ is not worse than $\boldsymbol{x}''$ in any of the function objectives;
2. $\boldsymbol{x}'$ is strictly better than $\boldsymbol{x}''$ with respect to at least one objective.

The set of nondominated solutions of the entire search space can be referred to as the Pareto front.

Pareto front definition

In the context of this work, a Pareto front is a set of solutions $\boldsymbol{x}$ that are all Pareto efficient towards the different outcomes from $f(\boldsymbol{x})$. These outcomes are a group of behavioral features $\boldsymbol{b} = b_1, b_2, \ldots$ (such as speed or torque output). These solutions show trade-offs between their performance towards these different features. A better performance can be expressed by a higher or lower value, depending on the feature.

3.3 NSGA-II

The nondominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002) is a multi-objective optimization EA that is based on the Pareto dominance relation. This second version of the algorithm improved its sorting algorithm, incorporated the concept of elitism, and dropped the requirement for a sharing parameter to be chosen *a priori*.

In each generation, the population is sorted in different fronts, or sets, according to the concept of non-domination (see Section 3.2.2). The first front includes the solutions that are non-dominant for the overall population, but dominate the solutions outside this set. The solutions for the second front are non-dominant between each other, are dominated by the solutions from the first front, and dominate the solutions from all the other fronts. This pattern is observed until the last front, which solutions are non-dominant between themselves, but are dominated by all the other solutions. Individuals from different fronts are assigned a different value of fitness (or ranks), with the ones from the first front having a rank of 1 (the best value), the ones in the second a rank of 2, an so on.

Each individual also has a crowding distance value associated, which measures how far it is from its set's neighbors. This value is the average Euclidean distance between the individual and the two nearest points to it, along the direction of each of the optimization objectives. The crowding distances comparisons are done only between

individuals in the same set. Larger crowding distances will promote diversity in the solution space of the population.

After sorting the population and calculating the crowding distances, individuals are selected for mutation and crossover. These will act as parents of new individuals, that will belong to the next generation. The selection is based, first, on the rank of the individual (*i.e.* the front it belongs to), and then on their crowding distances, where larger values are preferred. A binary tournament selection is used to decide which parents are matched for a comparison (Blickle and Thiele, 1996): two individuals are randomly selected each time, and the most adequate between them is selected as a parent.

Both the crossover and mutation operations results in a population partially composed of new solutions. The genetic operators used in the crossover and mutation phase are the simulated binary crossover (SBX) (Beyer and Deb, 2001), and the polynomial mutation (Raghuwanshi and Kakde, 2004). These operators are detailed in the next section.

The offspring solutions are then evaluated, and new fronts for the next generation of the population are created by choosing the individuals with the best fitness, both from the previous generation and the offspring solutions. Subsequent fronts are generated until the size of the population reaches the value selected by the user for the optimization process.

Algorithms 1, 2, and 3 show the main procedure of the NSGA-II optimization process, as well as some of its sub-functions. The next section details the SBX and polynomial mutation genetic operators. Figure 3.1 shows a flowchart of the optimization process.

3.3.1 SBX and polynomial mutation

SBX simulates the binary crossover observed in nature,

$$\boldsymbol{c}_{1,k} = \frac{1}{2}[(1-\boldsymbol{\beta}_k)\boldsymbol{p}_{1,k} + (1+\boldsymbol{\beta}_k)\boldsymbol{p}_{2,k}], \tag{3.1}$$

$$\boldsymbol{c}_{2,k} = \frac{1}{2}[(1+\boldsymbol{\beta}_k)\boldsymbol{p}_{1,k} + (1-\boldsymbol{\beta}_k)\boldsymbol{p}_{2,k}]. \tag{3.2}$$

Here $\boldsymbol{c}_{i,k}$ is the $\boldsymbol{k}^{\text{th}}$ component of the $\boldsymbol{i}^{\text{th}}$ child, $\boldsymbol{p}_{i,k}$ the $\boldsymbol{k}^{\text{th}}$ component of the selected parent, and $\boldsymbol{\beta}_k$ (≥ 0) a sample from a random number generator with the density

$$p(\boldsymbol{\beta}) = \frac{1}{2}(\boldsymbol{\eta}_c + 1)\boldsymbol{\beta}^{\boldsymbol{\eta}_c}, \text{ if } 0 \leq \boldsymbol{\beta} \leq 1, \tag{3.3}$$

$$p(\boldsymbol{\beta}) = \frac{1}{2}(\boldsymbol{\eta}_c + 1)\frac{1}{\boldsymbol{\beta}^{\boldsymbol{\eta}_c+2}}, \text{ if } \boldsymbol{\beta} > 1. \tag{3.4}$$

Here $\boldsymbol{\eta}_c$ is the distribution index for crossover, which determines how well spread the children are from their parents.

The polynomial mutation is encoded as

$$\boldsymbol{c}_k = \boldsymbol{p}_k + (\boldsymbol{p}_k^u - \boldsymbol{p}_k^l)\boldsymbol{\delta}_k, \tag{3.5}$$

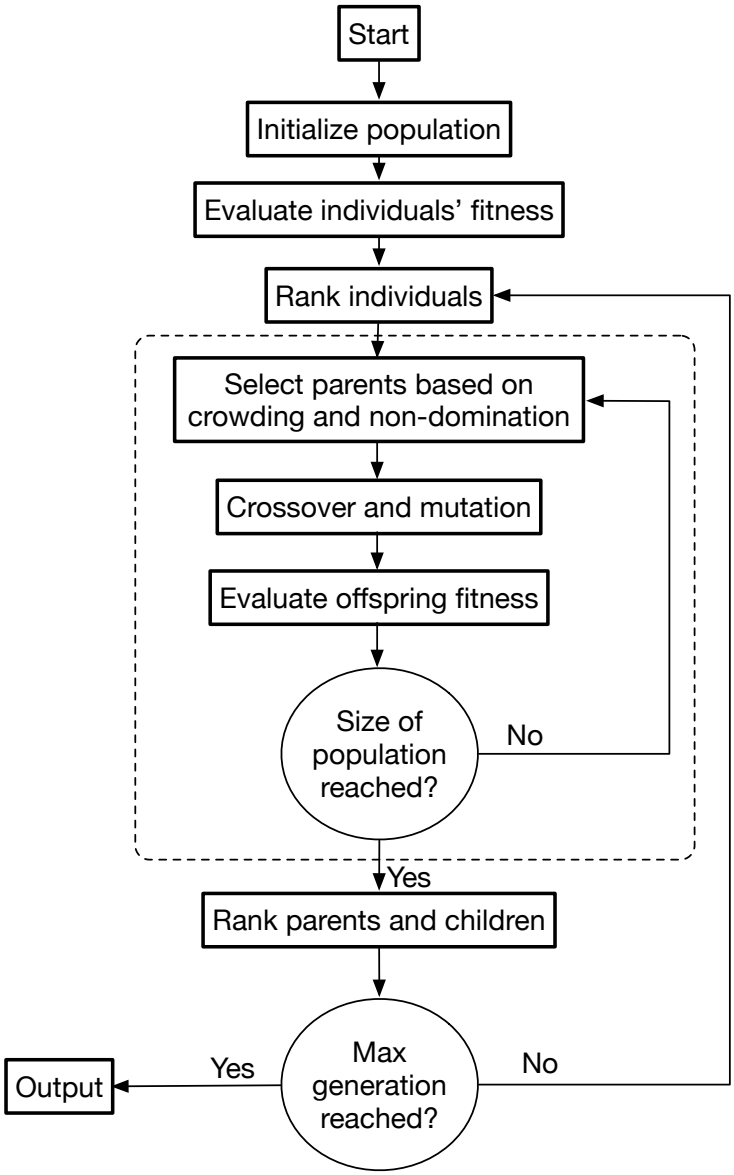


Figure 3.1: Flowchart of the optimization algorithm NSGA-II. Adapted from (Chang, Bouzarkouna, and Devegowda, 2015).

where $\boldsymbol{c}_k$ is the child individual, $\boldsymbol{p}_k$ the parent, and $\boldsymbol{p}_k^u$ and $\boldsymbol{p}_k^l$ are the upper and lower bounds on the parent $\boldsymbol{k}$ component. $\boldsymbol{\delta}_k$ is a small variation, calculated from a polynomial distribution

$$\boldsymbol{\delta}_k = 2(\boldsymbol{r}_k)^{\frac{1}{\eta_m+1}} - 1, \text{ if } \boldsymbol{r}_k < 0.5 \tag{3.6}$$

$$\boldsymbol{\delta}_k = 1 - [2(1-\boldsymbol{r}_k)]^{\frac{1}{\eta_m+1}}, \text{ if } \boldsymbol{r}_k \geq 0.5. \tag{3.7}$$

$\boldsymbol{r}_k$ is a random number between (0,1) with uniform sampling, and $\boldsymbol{\eta}_m$ is a mutation distribution index.

Algorithm 1 Pseudocode for the NSGA-II (Deb et al., 2002) procedure for the optimization of a given function. N is the population size, nr_generations the number of generations the optimization should run for, and $R_{mutation}$ and $R_{crossover}$, the rates, or probabilities, of mutation and crossover. Adapted from (Brownlee, 2011).

```
procedure FUNCTION_OPTIMIZATION(N, nr_generations, R_mutation, R_crossover)
    P ← INITIALIZE_POPULATION(N)
    fitness_P ← EVALUATE_POPULATION(P)          ▷ Evaluates individuals against the objective function.
    P ← NON_DOMINATED_SORT(P, fitness_P)
    P(d) ← CALCULATE_CROWDING_DISTANCES(P)
    selected_parents ← SELECT_PARENTS(P, P(d))
    children ← CROSSOVER_AND_MUTATION(selected_parents, R_mutation, R_crossover)
    for nr_generations do
        children_fitness ← EVALUATE_POPULATION(children)
        union ← MERGE(P, children)
        F ← NON_DOMINATED_SORT(union)                ▷ F is a collection of Pareto fronts.
        parents ← ∅
        F_L ← ∅                                      ▷ F_L is the last front of the sorting.
        for each F_i in F do                         ▷ Fronts are added to the next parents.
            if SIZE(parents) + SIZE(F_i) > N then    ▷ Is the population size exceeded?
                F_L ← F_i                            ▷ This is the last front added.
                BREAK                                ▷ Stop adding fronts.
            else
                parents ← MERGE(parents, F_i)        ▷ Merge front with the next parents.
            end if
        end for
        if SIZE(parents) < N then                    ▷ If N not reached, add solutions from F_L
            F_L ← SORT_BY_RANK_AND_DISTANCE(F_L)     ▷ Put the best ranked solutions first.
            for i ← 1 to N− SIZE(F_L) do             ▷ Add solutions until N reached.
                parents ← MERGE(parents, P_i)
            end for
        end if
        parents(d) ← CALCULATE_CROWDING_DISTANCES(parents)
        selected_parents ← SELECT_PARENTS(parents, parents(d))
        P ← children
        children ← CROSSOVER_AND_MUTATION(selected_parents, R_mutation, R_crossover)
    end for
    return children
end procedure
```

Algorithm 2 Pseudocode for the NSGA-II algorithm sub-functions for calculating crowding distances, and the parent selection process (Deb et al., 2002), both used in the main procedure.

function CALCULATE_CROWDING_DISTANCES(P)
  for each F_i in F **do**
    $n \leftarrow$ SIZE(F_i) ▷ n is the number of individuals in F.
    for $j \leftarrow 1$ to n **do** ▷ j indexes an individual of F_i.
      $F_i(d_j) \leftarrow 0$ ▷ distance for individual j in front F_i.
    end for
    for each objective function m **do**
      $I \leftarrow$ SORT(F_i, m) ▷ Sort individuals from F_i based on objective m [a].
      $I(d_1) \leftarrow \infty$ ▷ Assign infinite distance for boundary values
      $I(d_n) \leftarrow \infty$ ▷ for each individual in F_i.
      for $k \leftarrow 2$ to n **do** ▷ k indexes a population individual.
        $I(d_k) \leftarrow I(d_k) + \frac{I(k+1).m - I(k-1).m}{f_m^{\max} - f_m^{\min}}$ ▷ $f_m^{\max}$ is the max. value of the m^{th} objective function.
        ▷ $I(k).m$ is the value of the m^{th} objective function on the k^{th} individual in I.
      end for
    end for
  end for
end function
function SELECT_PARENTS(parents, parents(d)) selected_parents $\leftarrow \emptyset$
  for $i = 1$ to SIZE(parents)/2 **do** ▷ Binary tournament selection process.
    Randomly select p, q, from remaining parents
    if rank$_p$ < rank$_q$ **then**
      selected_parents $\leftarrow$ selected_parents $\cup\, q$
      Remove q from parents
    else if rank$_p$ > rank$_q$ **then**
      selected_parents $\leftarrow$ selected_parents $\cup\, p$
      Remove p from parents
    else ▷ rank$_p$ = rank$_q$
      if parents(d_p) > parents(d_q) **then** ▷ A larger crowding distance indicates a less desirable individual.
        selected_parents $\leftarrow$ selected_parents $\cup\, q$
        Remove q from parents
      else
        selected_parents $\leftarrow$ selected_parents $\cup\, p$
        Remove p from parents
      end if
    end if
  end for
  return selected_parents
end function

[a] This is equivalent to sorting the individuals based on features of locomotion, in the context of this work.

Algorithm 3 Pseudocode for the NSGA-II algorithm non-dominated sorting of individuals (Deb et al., 2002), used in the main procedure.

```
function NON_DOMINATED_SORT(P, fitness_P)
    for each individual p in P do
        S_p ← ∅                                  ▷ Set of all the individuals dominated by p.
        n_p ← 0                                  ▷ Number of individuals that dominate p.
        for each individual q in P do
            if p dominates q then
                S_p ← S_p ∪ q
            else
                n_p ← n_p + 1
            end if
        end for
        if n_p = 0 then                          ▷ No individuals dominate p.
            rank_p ← 1                           ▷ p belongs to the first front.
            F_1 ← F_1 ∪ p                        ▷ Add p to front 1.
        end if
        i ← 1                                    ▷ i is a front counter.
        while F_i is not empty do
            Q ← ∅                                ▷ Set to store the individuals for F_{i+1}
            for each p in Fi do                  ▷ Cycle through the individuals in the current front.
                for each q in S_p do             ▷ Check the individuals dominated by p.
                    n_q ← n_q − 1                ▷ p is no longer accounted for.
                    if n_q = 0 then              ▷ q is no longer dominated by any individual.
                        rank_q = i + 1           ▷ No subsequent fronts would dominate q.
                        Q ← Q ∪ q                ▷ q belongs to F_{i+1}.
                    end if
                end for
            end for
            i ← i + 1                            ▷ Increment the front counter.
            F_i ← Q
        end while
    end for
end function
```

Chapter 4

A general framework for biped locomotion control optimization

In this chapter a framework for the optimization of humanoid locomotion controllers is proposed. It can be applied to any controller with open parameters, and any humanoid robot that can achieve locomotion through its control. The method should be able to optimize towards different locomotion features (*e.g.* speed, stability measures), and towards different environment changes, such as terrain changes or alterations to the robot itself. Finally, it should be able to automatically select the more relevant parameters to optimize, in order to make the process faster and more efficient.

The framework is based on derivative-free optimization. This choice is based on the fact that the task itself is complicated, which makes a model's derivative hard to obtain. Chapter 2 explored the complex dynamics of locomotion, and changes in the robot and its environment that lead to this issue. Furthermore, obtaining the necessary derivative information for every controller and robot is unfeasible, which is a requirement for this framework, since it is intended to be applicable to any combination of humanoid robot and locomotion controller with open parameters.

This chapter begins by delineating the overall problem and related mathematical notation, then describes the general approach used to tackle it, and follow that by specifying each component of the framework, as well as considering the options available to perform each task. After that, an approach for a sensitivity analysis on the environment parameter is described, along with a correlation analysis that can be used to choose the most appropriate parameters to tune from a larger list.

4.1 Problem definition

Consider a humanoid locomotion controller with open parameters, *i.e.*, parameters whose values are not set. Each set of values for these parameters are a tentative solution $\boldsymbol{x}$ for the task of walking, that, when applied to the control of a robot, will

result in observed behavioral features of the locomotion $\boldsymbol{b}$ (*e.g.* speed, applied torque),

$$\boldsymbol{b} = f(\boldsymbol{x}). \tag{4.1}$$

f being the mathematical function between the inputs ($\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$) and the outputs ($\boldsymbol{b} = \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots$). These solutions can also be evaluated in terrains with some of their characteristics modeled by parameters (*e.g.* friction, slope) $\boldsymbol{\theta}$,

$$\boldsymbol{b} = f(\boldsymbol{x}, \boldsymbol{\theta}). \tag{4.2}$$

The goal is to find the solution $\boldsymbol{x}^*$ that results in the best possible value that can be found for one of the observed features (*e.g* $\boldsymbol{b}_1$, the locomotion speed). When multiple features are considered, we try to find the set of solutions $\mathcal{X}^*$ that result in the best values for these features. When a solution that results in the best values for every feature simultaneously cannot be found, the goal is to obtain one that results in varying degrees of performance trade-off (*e.g.* trading speed for energy efficiency). Different values of $\boldsymbol{\theta}$ will affect the outputs of the system, and, consequently, the best solutions found in these optimizations.

4.2 Exploration framework

When applying derivative-free optimization to this problem, several trials with different controller parameters and, possibly, terrains or other environmental changes need to be conducted. This process, which is referred to as the exploration phase (or the training phase, in the larger context of the adaptation framework), leads to obtaining a set of the solutions used, $\mathcal{X}_\mathrm{t}$, which is associated with a set of the environment parameters, Θ_t, and a set of the behavioral features, $\mathcal{B}_t$,

$$\mathcal{B}_t = \{f(\boldsymbol{x}, \boldsymbol{\theta}) \mid \boldsymbol{x} \in \mathcal{X}_\mathrm{t}, \boldsymbol{\theta} \in \Theta_\mathrm{t}\}, \tag{4.3}$$

that resulted from the training simulations of those solutions $\boldsymbol{x}$ on the terrain defined by the chosen $\boldsymbol{\theta}$. These results are stored in a dataset $\mathcal{D}_\mathrm{t}$,

$$\mathcal{D}_\mathrm{t} = \{\mathcal{X}_\mathrm{t}, \Theta_\mathrm{t}, \mathcal{B}_\mathrm{t}\}, \tag{4.4}$$

which will be called the training dataset, in the context of the overall adaptation approach.

From this dataset, one can select a single solution that maximizes (*e.g.* speed), or minimizes (*e.g.* torque output), a given behavioral feature. To select solutions that are optimal towards multiple features, the concept of Pareto optimality (see chapter 3) can be used. The set of solutions that are all Pareto optimal can be referred to as the Pareto front, and this dataset as $\mathcal{D}_\mathrm{Pf}$. Figure 4.1 shows an overview of this optimization approach.

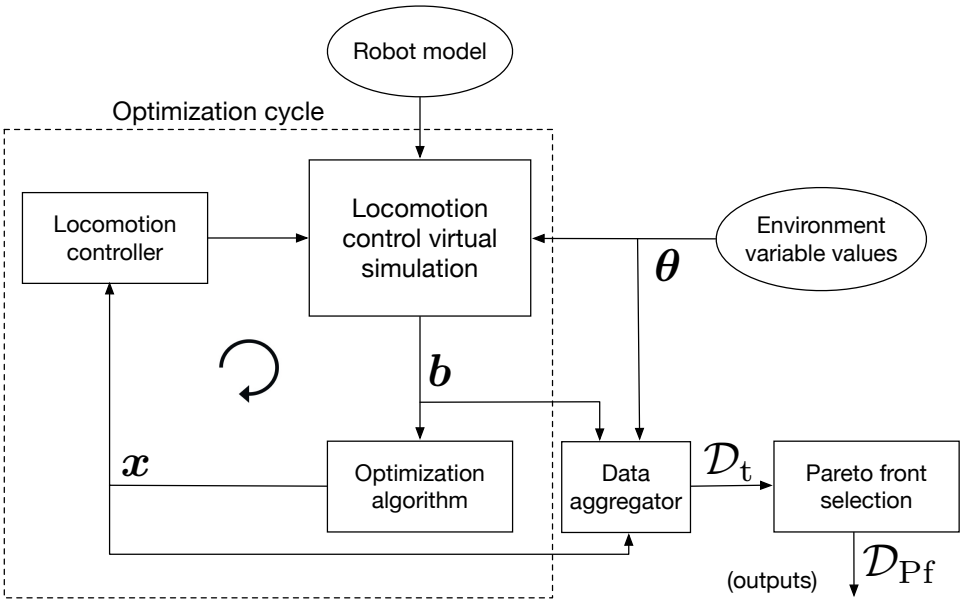


Figure 4.1: Architecture of the exploration/training framework. The robot model and the environment, along with a control structure defined by its open parameters being tuned ($\boldsymbol{x}$), are fed to the virtual simulation of the locomotion process. The optimization algorithm selects a set of parameters values that will be tested, taking into account the optimization of the features ($\boldsymbol{b}$) received from the virtual simulation at the end of each trial. This is a cyclic process, which is ended by the optimization algorithm after a certain number of trials, or when certain conditions for the optimization are met. The process can be repeated for different values of the environment variable ($\boldsymbol{\theta}$). The different solutions $\boldsymbol{x}$, resulting features $\boldsymbol{b}$, and environment values $\boldsymbol{\theta}$ are aggregated into a single set. The end result is the dataset $\mathcal{D}_{\mathrm{t}}$ defined in Equation 4.4.

4.3 Selecting the components of the framework

The proposed framework is intended to be as general as possible in terms of accepting which control architecture will be optimized (and its corresponding parameters), the locomotion behavioral features optimized for, the humanoid robot being controlled, the terrain/environment aspect modeled, the program used to simulate the locomotion trials, and, to some extent, the optimization algorithm. These have to work with each other in a compatible manner. During the course of this section the possibilities of choice for each of these components will be discussed, as well as highlighted the ones that were part of the experiments conducted.

4.3.1 Robot model

This framework is intended to be applied to a humanoid robot, generally having a head, a torso, two arms, and two legs. Some of these robots may only have the lower or upper part of the body, and since this is intended to be used in the optimization of bipedal locomotion only the former are applicable. From an implementation standpoint, the robot must be compatible with the locomotion controller, and a model must be available or constructed for the simulator.

The robots used in this work were models of the iCub and DARwIn-OP robots, and a virtual robot called the XDE-manikin. All of them are bipedal, and described in Appendix A.

4.3.2 Locomotion controller

Any humanoid locomotion controller should be applicable, as long as it has open parameters that need to, or can, be tuned. Sections 2.2.1 and 2.2.2 contain discussions on some of the most relevant control approaches in this context, and highlight how common it is for them to have open parameters.

In the situations where there are parameters that can be tuned, the question becomes whether they affect the locomotion behavior in a way that is meaningful for the user. Some controllers, like the dynamics based one used with the iCub and the XDE-manikin robots, described in Appendix B, have open parameters that directly correlate to locomotion characteristics, such as controlling each step's length, or its height. When using other controllers, such as the CPG based one applied to the DARwIn-OP robot (see Appendix C), the parameters affect the locomotion outcome in ways that are hard or impossible to predict without conducting experiments.

The optimization framework presented collects thousands of outcomes of different solutions (parameter sets), providing the necessary information to analyze the effect of these parameters in the locomotion aspects one is interested in. A concrete way to do this analysis was developed and is formalized in Section 4.5.

4.3.3 Environment variable

The variables that would be interesting, or useful, to model and optimize for are the ones that most impact the locomotion behavior and/or are harder to adapt to via other methods. These can be properties of the terrain the humanoid is walking on, or of the robot itself, for example.

In a given terrain, a robot's foot can exert a given amount of horizontal friction on the floor, before it causes the robot to slip. The value that represents that threshold is a common and potentially impactful variable (Kajita et al., 2004; Hobon, Elyaaqoubi, and Abba, 2013) that affects the locomotion process of every biped. The coefficient of friction (CoF) — the ratio between the tangential ($\boldsymbol{ft}_c$) and normal ($\boldsymbol{fn}_c$) forces acting on the contact points of the feet with the ground — serves as a measure of a limit for when this potential displacement can start to occur,

$$\text{CoF} = \frac{\boldsymbol{ft}_c}{\boldsymbol{fn}_c}. \tag{4.5}$$

As discussed in Chapters 1 and 2, what happens during the contact of a biped robot with the floor is both instrumental to cause the movement of the system, and dangerous for its stability, if not done with the necessary precautions. The horizontal forces that can be applied to the contact points differ from floor to floor, and that affects the behavior one can achieve.

Another important property is the slope of the terrain when walking up, or down, a ramp (B et al., 2012; Dong, Zhao, and Zhang, 2011). It affects the way the robot has to direct its whole body, which changes its attitude, and, consequently, its stability. This constrains the possible length of each step, and may easily lead to a fall.

One last possible terrain related property to model is its impedance — a measure of its resistance to motion. In many simulations the floor is considered as a completely rigid body, but allowing it to be temporarily deformable, even to a small extent, would lead to changes in the way the impact of the robot's feet with the ground affects the robot, and consequently the locomotion process.

A usual approach to terrain changes or obstacles in biped locomotion research is to make the controller adaptable to uneven terrain (Morisawa et al., 2014; Asif and Iqbal, 2011; Manko, 1992) (other works were discussed in Section 2.2.4). This can include terrain with obstacles, terrain with smoother or rougher patches, or sudden height or slope changes, for example. Adaptation of a control framework to this situation tends to be more complex and perception related, requiring ways to deal with multiple sub-problems whose solutions may vary for different control approaches. We do not look to adapt to these specific situations, and this framework is not appropriate for it. Variables like a CoF or the slope are more universally present and impactful.

Changes that come from the robot itself could also be encapsulated by an environment variable. In Chapter 5, for instance, optimizations were conducted for the same robot model, but with varying height and mass values, and the body proportions being kept intact. Cully et al. (2015) proposed an adaptation framework that they applied to a hexapod robot with a damaged leg.

Exploring the way these changes affect the humanoid's locomotion can provide insight to what degree they require the need for different control approaches. These can be needed to maintain or optimize the task performance observed before the changes.

4.3.4 Optimized locomotion features

The features of locomotion that are defined, and used as optimization objectives, need to be chosen with the user requirements and robot stability in mind. They should be common to the largest amount of bipedal robots possible (as in the possibility of measuring them), and relevant, either in terms of performance, or stability. Ideally, they should use no sensors, or common ones (i.e. feet sensors as opposed to vision). Performance oriented features such as speed can be combined with criteria directed for stability, such as the position of the system's CoP.

Performance indicators

When it comes to performance, some of the most common needs are greater speeds and energy efficiency. Energy can be linked to the torque output of the robot's joints throughout the locomotion task, and some measure of this as a single value, such as the total torque output,

$$\tau_{\text{total}} = \int_0^{t_\text{f}} \boldsymbol{\tau}_t^\top \boldsymbol{\tau}_t \, dt, \tag{4.6}$$

or its average,

$$\tau_{\text{mean}} = \frac{h}{t_\text{f}} \sum_{t=0}^{t_\text{f}} \boldsymbol{\tau}_t \boldsymbol{\tau}_t. \tag{4.7}$$

Here t_f refers to the total duration of the simulation, $\boldsymbol{\tau}_t$ is a vector containing the current torque of every joint at a given time t, and h is the time step of the simulation. Any of these calculated quantities can be minimized to promote lower energy usage.

The overall error of the locomotion trajectory can also be measured as a total throughout the simulation process, or as an average,

$$\text{error}_{\text{mean}} = \frac{h}{t_\text{f}} \times \sum_{t=0}^{t_\text{f}} \text{distance}(\text{trajectory}_t, \text{robot\_position}_t). \tag{4.8}$$

Here distance(.) is the Euclidean distance between the 3D points trajectory_t, and robot_position_t, which are the robot's pretended location according to the locomotion trajectory, and its actual location, at time t. This feature can be used to promote both stability and performance. When walking in a straight line the trajectory error is complementary to the speed, since deviations from that path will decrease the distance achieved in that direction. It can promote stability by lowering lateral overextensions of the CoM that can cause balances issues, which are ideally not present in the trajectory.

Stability

Stability related criteria can be any measure that optimize the locomotion towards balanced movement and, fundamentally, the robot not falling. Representing a fall as a binary variable may not be ideal for the optimization algorithm, but adding a penalty for some of the objectives in the case one occurs can discourage the algorithm from exploring similar solutions, since these types of algorithms do not focus on the solutions that show bad results.

Indicators related to the friction between the contact points and the ground can be measured with the use of force sensors on the feet. As previously discussed in this chapter, the ratio between the tangential and normal contact forces can be used to evaluate the risk of slipping due to lack of friction. Minimizing these tangential forces, or just trying to keep them from reaching a certain threshold, can increase the overall stability of the task (Hurmuzlu, Génot, and Brogliato, 2004; Zhou et al., 2013).

The position of the center of pressure is important for dynamic balance, as discussed in Section 2.1.1. Indicators that help keep the CoP in the support polygon of the biped are therefore useful. Simply minimizing its distance to the center of the polygon is an option, but this may also keep the robot from using behaviors that are riskier but more efficient towards performance indicators. Instead minimizing the time spent by the CoP outside the support polygon, without care of its distance from the edges, can allow for higher speeds, for example, but it comes with the possible downside of lower stability.

Asymmetries in human and humanoid locomotion are generally linked to irregular, and perhaps inefficient, patterns (Hyon and Emura, 2005; Handžić and Reed, 2015). The time spent by each foot in the ground in relation to the total time for the locomotion can be referred to as the duty factors. Having similar values of duty factor for both feet is observed in symmetric gaits. The difference between those duty factors can, therefore, be a useful quantity to minimize. On the other hand, some studies suggest there may be performance benefits and even a biological basis for asymmetrical locomotion patterns. Gregg et al. (2011) suggested that asymmetric high-speed gaits can be more stable than symmetric gaits when certain changes in the environment are observed. In another study Gregg, Dhaher, and Lynch (2011), they discuss the hypothesis that, in human and humanoid locomotion, each leg may have different overall roles, such as support and propulsion, resulting in functional asymmetry.

This possibility of the chosen indicator (minimizing the difference between duty factors) not being appropriate to a better performing locomotion process, can happen with any feature. One way to solve this would be to address them individually, carefully researching and analyzing the implications of optimizing towards a given behavior. Ideally, especially for this framework, one would have a way to measure how an indicator affects the task outcome, regardless of what that indicator measures, or what it is being maximized, or minimized. This can be achieved by applying the correlation analysis procedure described in Section 4.4.

4.3.5 Simulator

A physics based simulation of the locomotion process is necessary to speed up the process of optimization, which requires thousands of simulations to acquire a repertoire large enough to support meaningful analysis of the effects of each parameter, and the adaptation process introduced later. The simulator also needs to support the implementation of the controller and the robot model, as well as the modeling of the intended environmental parameters.

When choosing which simulators to use during a project like this, availability, price, and the time required to learn how to work with the tools are important aspects. The iCub and the XDE-manikin were simulated using XDE, a simulator developed by CEA-LIST (Merlhiot et al., 2012) that enables the simulation and control of physically interacting mechanisms. XDE is not an open source engine, but it was available at one of the research laboratories this work was developed in, ISIR. There was also a model of the iCub available in the toolkit, as well as an implementation of the dynamics based control system used along with it. For similar reasons, the Webots simulator (Michel, 2004) was used to work on the DARwIn-OP experiments. It is a commercial robot simulator developed by Cyberbotics, often used for research purposes, and it uses the Open Dynamics Engine (ODE) (*ODE Wiki*) for 3D rigid body dynamics. It was available in the other research laboratory this worked was developed in (ASBG), and includes a model of the DARwIn-OP robot, in addition to ASBG possessing an implementation of the CPG based locomotion controller that was adapted for our experiments.

4.3.6 Optimization algorithm

The optimization framework is intended to be as broad as possible, which means the optimization algorithm should not rely in any analytical support such as derivative information, since that would be case specific to different controllers and robots. Additionally, the algorithm should also support multi-objective optimization. This will lead to greater flexibility when it comes to users' requirements towards the locomotion process, since one would be able to choose between prioritizing different features. A multi-objective optimization would also result in a repertoire of solutions with greater potential to perform well in different terrains, and, consequently, to adapt to them.

Evolution strategies and evolutionary algorithms

Some of the most popular derivative-free optimization approaches are evolution strategies (ES), such as the covariance matrix adaptation evolution strategy (CMA-ES), and evolutionary algorithms (EA), such as genetic algorithms (GAs). Both belong to the overall field of evolutionary computation, and are based on biological evolution. These use algorithms that are based in multiple generations of a population of a set of solutions, referred in this context as the parent solutions. These parent solutions are mutated and combined to create new solutions. Some of the population individuals are then selected, based on their fitness towards an objective function value, to become

the parents in the next generation of the population. This ideally leads, over various generations, to individuals with better and better fitness values. Section 3.2 provides more details into EAs.

CMA-ES (Hansen, 2006) updates the covariance matrix that represents the dependencies between the variables of the distribution that represents a candidate solution. Igel, Hansen, and Roth (2007) proposed a multi-objective CMA-ES, highlighting that it shows invariance properties that imply uniform performance on a class of objective functions.

Another example of a multi-objective EA (MOEA) is the non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002) that improves a population of candidate solutions to a Pareto front. It was designed to alleviate the issues of computational complexity and the lack of elitism that this type of algorithms usually possesses. Elitism consists in keeping a small group of the fittest candidates unchanged into the next generation, which can prevent losing good solutions and speed up the optimization process (Zitzler, Deb, and Thiele, 2000). NSGA-II also does not require the need to specify a sharing parameter, which is used as a way to ensure diversity in a population, providing an alternative to the problematic. Deb, Mohan, and Mishra (2005) approached the problem of the trade-off between having a converged and well-distributed Pareto front, and the computational time of the optimization. They proposed a MOEA intended to be a compromise between convergence near the Pareto front, diversity of solutions, and computational time, and named it ϵ-MOEA, because of its ϵ-dominance concept. A solution dominating relative to another one has to perform better than it in every objective, and ϵ-dominance (Laumanns et al., 2002) guarantees that the difference between those solutions is of a factor of at least ϵ, thereby promoting diversity in the population.

Particle Swarm Optimization

Particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), similarly to EAs, uses a population of candidate solutions (a swarm of particles), which are moved around a search space towards better solutions according to the particle's position and velocity. The particle's movement is informed by both its local best known position and the best known position in the whole search-space. Dai, Wang, and Ye (2015) proposed a multi-objective PSO algorithm with the diversity of solutions in mind. Their approach decomposes the objective space of the problem in a set of sub-regions, which are then made to have a solution to maintain diversity. They compared it to, among other algorithms, NSGA-II, finding it to be better in terms of convergence and diversity.

MAP-Elites

Considering the problematic of balancing reaching optimal solutions with the diversity observed in the Pareto front, an algorithm can be mostly concerned with the former, to the point of being considered a search algorithm. Such is the case of the multi-dimensional archive of phenotypic elites (MAP-Elites) (Mouret and Clune, 2015), that

produces a repertoire of solutions intended to be diverse in the feature space (*e.g.* speed, torque output). The dimensions of variation of the feature space can be defined by the user, and the algorithm illuminates high performing solutions in that space, allowing one to better understand the relation between some of those features and selected performance criteria.

Software implementation: Sferesv2

Also important in the choice of an optimization algorithm is how easy it is to be implemented, and how easy to use that implementation is. During this work we used Sferesv2 (Mouret and Doncieux, 2010), a high-performing and lightweight evolutionary computation framework written in C++. It contains implementations of multiple EAs, both for single and multi-objective optimization.

4.4 Sensitivity analysis on the environment variable

In the subject of optimizing in a given environment/context, a particular issue is knowing if solutions optimal in that context extrapolate well to other terrains. That is, if those solutions will result in good performances in contexts different from the ones they were optimized for. To gather insight regarding this hypothesis, a sensitivity analysis can be conducted. It has the purpose of analyzing how the uncertainty caused by varying the environment variables affects the behavior of solutions that were found optimal, or near-optimal, in a specific environment.

A schematic of the sensitivity analysis process can be seen in Figure 4.2. From the exploration phase, various datasets are obtained $\mathcal{D}_{\mathrm{t}_1}, \mathcal{D}_{\mathrm{t}_2} ... \mathcal{D}_{\mathrm{t}_n}$, one for each $\boldsymbol{n}$ values of terrain parameters tested ($\Theta_{\mathrm{t}} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 ... \boldsymbol{\theta}_n\}$). From each dataset, the Pareto front is extracted ($\mathcal{D}_{\mathrm{Pf}_1}, \mathcal{D}_{\mathrm{Pf}_2} ... \mathcal{D}_{\mathrm{Pf}_n}$), and the solutions from these are then combined into one single list $\mathcal{X}_{\mathrm{Pf}_{\mathrm{all}}}$. The next step is to test all the solutions from $\mathcal{X}_{\mathrm{Pf}_{\mathrm{all}}}$ in different terrain parameters Θ_{s} (which could simply be equal to Θ_{t}, or extend this set) in order to obtain a dataset that can be used for the sensitivity analysis. This is defined as $\mathcal{D}_{\mathrm{s}} = \{\mathcal{X}_{\mathrm{Pf}_{\mathrm{all}}}, \Theta_t, \mathcal{B}_{\mathrm{s}}\}$, with $\mathcal{B}_{\mathrm{s}} = \{f(\boldsymbol{x}, \boldsymbol{\theta}) \mid \boldsymbol{x} \in \mathcal{X}_{\mathrm{Pf}_{\mathrm{all}}}, \boldsymbol{\theta} \in \Theta_{\mathrm{s}}\}$.

Each Pareto front of solutions contained in the dataset $\mathcal{D}_{\mathrm{s}}$ can then be analyzed in relation to indicators such as their average success rate across all environments tested, or the overall averages of each performance indicator. These indicators are useful in understanding how much the change in environment changes the performance of the sets of solutions.

4.5 Correlation analysis

Information about the effect the tuned parameters have in the outcome of the selected locomotion features, and the correlation they have between themselves, can give the user important insight when it comes to select the controller parameters (used as opti-

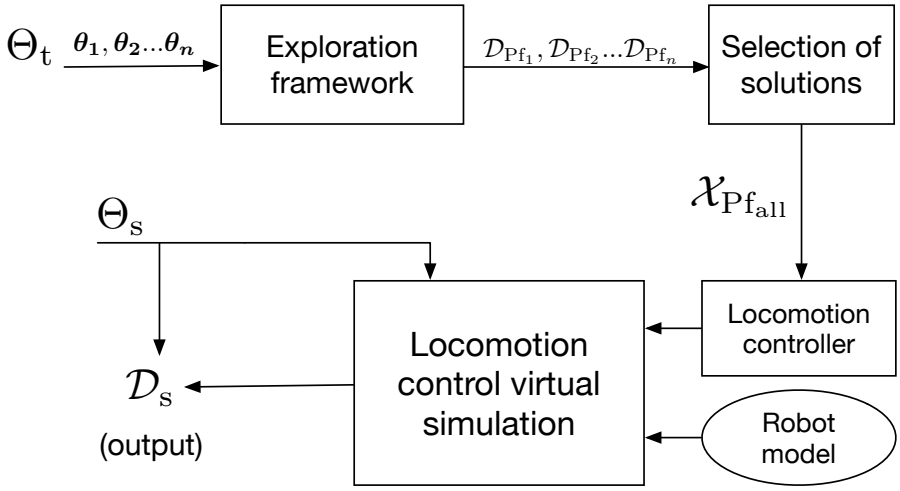


Figure 4.2: Schematic of the process of sensitivity analysis towards the terrain variable. The exploration framework is shown in Figure 4.1. The robot model, locomotion controller, and the locomotion control virtual simulation are the same used in the exploration process. From the Pareto fronts datasets ($\mathcal{D}_{\mathrm{Pf}}$), all the solutions optimal in the previous optimizations are extracted ($\mathcal{X}_{\mathrm{Pf_{all}}}$). These solutions are then used to configure the locomotion controllers tested for all the context values in Θ_{s}. The behaviors observed in these trials are collected together with these last tests in the sensitivity dataset, $\mathcal{D}_{\mathrm{s}}$.

mization inputs) that should be used in later optimizations. Parameters that result in controller with very low impact in the speed of the robot, for example, can be discarded.

This correlation analysis consists in calculating, for each parameter, its correlation with the other parameters and with each locomotion feature (the optimization objectives). On top of this, the coefficient of variation (cv) of each parameter were also determined. This coefficient is defined as

$$\text{cv} = \frac{\boldsymbol{\sigma}}{\boldsymbol{\mu}} \times 100\%, \tag{4.9}$$

where σ is the standard deviation of the given parameter distribution, and μ is its mean, and is used as a measure of the dispersion of the parameters. Correlations are calculated with the Pearson product-moment correlation coefficients (Pearson, 1895), and measure the correlation between two variables. They are defined as

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} \times C_{jj}}}, \tag{4.10}$$

where C is the covariance matrix, and C_{ij} represents the covariance between the variables with indexes i and j, which measures how much these variables change together. C_{ij} represents the variance of the variable when $i = j$. A value of 1 or -1 for this coefficient means there is a linear correlation between the variables, positive or negative. A value of 0 means there is no correlation.

4.5.1 Tuning the optimization setup

The results of this analysis can be used to decide which parameters and features are used in later optimization processes. The way these are applied depends on a variety of factors related to the user's requirements, such as how many parameters one wants to keep in the optimization process, how valued some of the features are as performance indicators, and the weight given to the correlation between parameters, objectives, and between both of these.

The number of parameters may be reduced to a predetermined number, with the threshold for choosing them being set accordingly, or the threshold can be predetermined, with the number of parameters remaining depending on how many of them respect it. The features, or optimization objectives, may also be filtered according to this analysis. Some of these are considered performance indicators, such as the speed of locomotion, and the correlation of other features with these can indicate their impact in the optimization process. The performance indicators themselves are desirable to keep, since they represent the requirements for the task being optimized.

The analysis calculates the correlation between all the parameters and features of the optimizations, but only a few values are used in the decision stage. The average between a parameter and the performance indicators, for example, can give insight in which parameters have a greater effect in the locomotion performance. The average correlation between a given parameter and the rest of the parameters is also important. Parameters that show a good correlation between each other may be superfluous to the

optimization process, since they are changed at similar rates. This can be further explored by looking at each parameters' correlation with the performance indicators. If those values are similar, they may serve a similar role in the locomotion process, and therefore be redundant in the process. A balance between the weight given to the parameters effect on the performance indicators and this potential redundancy must be decided on, in order to make decisions that result in a faster and more efficient optimization.

Chapter 5

Optimizing the control of biped locomotion in different conditions

This chapter details the experiments regarding the optimization of locomotion controller's in different conditions, applying the framework presented in Section 4.2 and shown in Figure 4.1.

The experiments conducted are the optimization of the locomotion control of a model of the iCub robot in floors with different frictions, a virtual manikin humanoid with different sets of height and mass combinations, and a model of the DARwIn-OP robot on floors with different slopes.

These experiments have a similar setup. The optimization algorithm NSGA-II is used to tune the locomotion controllers in different conditions, using different parameters as optimization inputs, and different locomotion features as optimization objectives. These are repeated in different contexts, and a sensitivity analysis can be conducted to analyze how different contexts result in different behaviors when using the same controller solutions. Additionally, a correlation analysis can be used to gather information about the effect different parameters have in the locomotion features, which can be used to tune the optimization setup towards more efficiency.

5.1 Locomotion control of the iCub on floors with different frictions

The experiments presented in this section are optimizations of a locomotion controller applied to an iCub model. This controller is based on dynamics modeling, and uses a hierarchy based task system. At different stages, experiments with different purposes are conducted in this context. In the first stage, different sets of parameters are tuned towards different objectives, in order to explore the best setup for optimizing towards the performance criteria. Afterwards, the optimization process is extended to cover floors with different values of friction, resulting in multiple optimal sets of solutions, one toward each terrain optimized for. Finally, a sensitivity analysis is conducted on

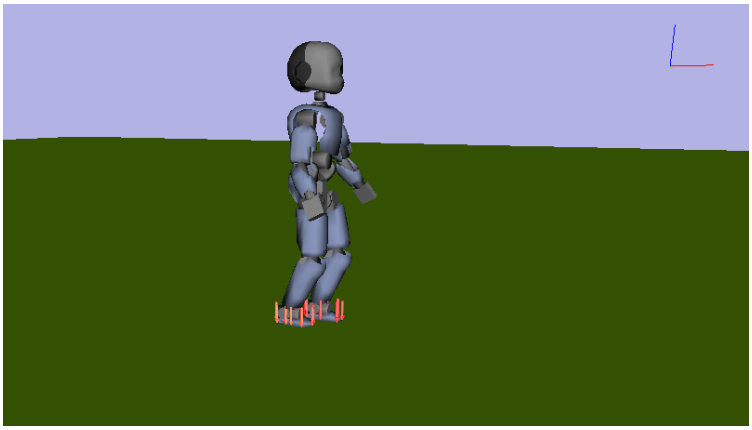


Figure 5.1: Screenshot of the iCub during simulation in the XDE framework.

these last sets of results, in which the performance of solutions optimized for one friction value is evaluated in other terrains, which these solutions are not optimized for.

5.1.1 Optimization framework setup

The robot model involved in these experiments is the iCub, a humanoid robot with dimensions similar to those of a 3.5 years old child (Parmiggiani et al., 2012), described in Appendix A, Section A.1. The controller used is the dynamics based controller from Salini described in Appendix B, and the trials were simulated in XDE (described in Section 4.3.5). Control parameters are tuned towards multiple locomotion features, with the optimization being conducted by a state of the art EA called NSGA-II, described in Section 4.3.6, included in the evolutionary computation C++ framework Sferesv2 (Mouret and Doncieux, 2010). Figure 5.1 shows the model of the iCub and the floor used in the XDE simulations, while Figure 5.2 shows a scheme of the optimization setup used in the iCub's trials.

Table 5.1: Values used for the NSGA-II parameters used in the Sferesv2 evolutionary computation framework.

Parameter	Value
Mutation type	Polynomial
Crossover type	SBX
Mutation rate	0.1
Crossover rate	0.5
η_m	15.0
η_c	10.0
Population size	100
Number of generations	100

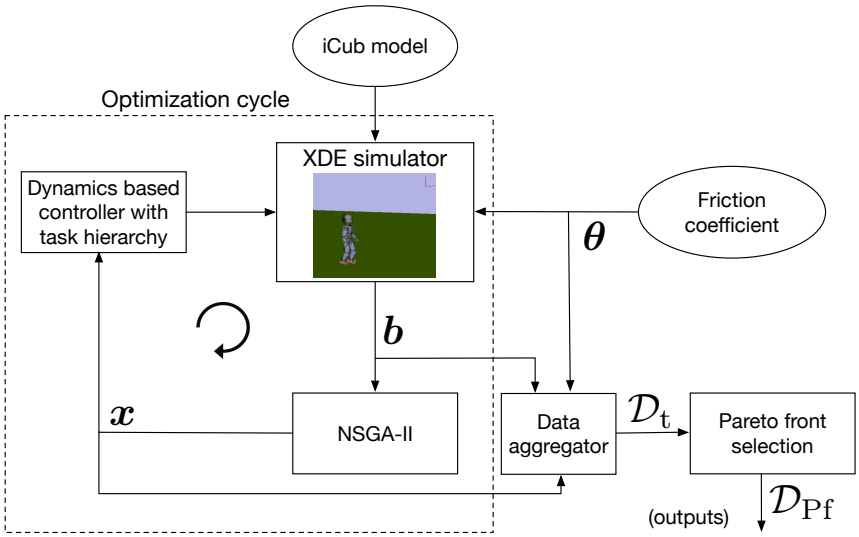


Figure 5.2: Architecture of the optimization process for the iCub. $\boldsymbol{x}$ is the vector of values for the parameters being optimized, $\boldsymbol{b}$ is the vector of locomotion features results from a trial, $\boldsymbol{\theta}$ is the context variable, and $\mathcal{D}_t$ and $\mathcal{D}_{pf}$ are datasets containing all the data from the optimizations, and only the data related to the Pareto fronts of the optimizations, respectively. Refer to Figure 4.1 for a description of the framework.

Table 5.1 shows the values used for the NSGA-II related parameters in all of the experiments with the iCub. The mutations serve to maintain genetic diversity, and, in this case, a polynomial distribution is used to perturb the solutions from generation to generation (Deb, 2001). In crossover, multiple parent solutions are used to produce a child solution for a new generation. SBX, or the simulated binary crossover (Deb, 2001), simulates the exchange between solutions as an exchange of bits between binary coded floating point numbers, without actually having to use a binary representation. Having a higher mutation rate usually promotes exploration of new solutions, while a higher crossover rate tends to have the algorithm converge to specific good solutions, in what is called exploitation (Črepinšek, Liu, and Mernik, 2013). A trade-off between these behaviors must be sought, in order to avoid the search to spread too much with exploration, and not find good solutions, or, on the other side, an early convergence to local optima with too much exploitation, making it less likely to find global optima. η_m is a parameter related to the polynomial mutation, with smaller values resulting in stronger mutations. Similarly, η_c is related to SBX, and a larger value creates solutions closer to the parents, with smaller values creating more distant solutions.

Experiments that optimize the control both for a single type of terrain and for terrains with different friction coefficients are described ahead. These are referred to as the stage 1 and stage 2 experiments. In stage 1, three separate tests are conducted, which are referred to as 1-A, 1-B, and 1-C.

5.1.2 Stage 1: Locomotion control optimization

These optimizations were performed in the same terrain, with the iCub being optimized to walk forward, in a straight line, for 20 s. NSGA-II runs for 100 generations, with a population size of 100, resulting in 10000 trials. The tests differ on the parameters tuned and the locomotion features optimized for. Runs which ended with a physics or/and controller failure, resulting in a robot fall and/or explosion [1] where given fitness penalties of 0 m/s for the mean speed, and 99999 $N^2.m^2$ for the joint torque output indicator. This last number only needs to be higher than the highest values of torque output observed, keeping it out of the Pareto front.

Parameters

The parameters involved in the optimization process (see Table 5.2) were the length of the robot's steps, their width, their height, their duration, the ratio between the duration of the single and double support phases, and two ZMP related parameters: RonQ, and the ZMP horizon. RonQ is a computational related quantity that controls the ratio between the tracking of the control vector and the tracking of the state vector. The ZMP horizon is the time horizon of the prediction made by the ZMP dynamics simplification used to preview the state of the ZMP and the CoM of the system. All seven of these parameters were tuned in stage 1-A, with RonQ being dropped in the later stages.

Table 5.2: Range of values used for the parameters involved in each optimization process of the first stage of the iCub optimization. Empty cells signal that the respective parameter was not tuned in that experiment.

	Parameter's range		
Parameter	1-A	1-B	1-C
Step length (m)	0.07 - 0.15	0.07 - 0.138	0.07 - 0.138
Step width (m)	0.04 - 0.15	0.05 - 0.10	0.05 - 0.10
Step height (m)	0.003 - 0.037	0.005 - 0.0395	0.005 - 0.0395
Step duration (s)	0.5 - 3.5	0.5 - 3.0	0.5 - 3.0
Step phase ratio	0.5 - 0.95	0.525 - 0.94	0.525 - 0.94
ZMP horizon	1.025 - 2.0	1.025 - 2.0	1.025 - 2.0
RonQ	0 - 0.00027	—	—

Performance related locomotion features

The locomotion features optimized for in 1-A are the mean speed ($\boldsymbol{v}_{\text{mean}}$, maximized) of the robot and a torque indicator (minimized) that consists in the sum of all the

[1] Here explosion refers to a simulator physics' failure where all the robot's parts rapidly separate from each other, projected in different directions.

squared torque output throughout the locomotion,

$$\boldsymbol{\tau}_{\text{total}} = \int_0^{t_\text{f}} \boldsymbol{\tau}_t^\top \times \boldsymbol{\tau}_t \, dt, \tag{5.1}$$

with $\boldsymbol{\tau}_t$ being the vector of joint torque outputs at time t, and t_f the total time of the experiment (20 s). These two indicators are also the performance indicators for all the experiments of the first stage.

Experiment 1-B stability related locomotion features

In experiment 1-B two locomotion features are added: the maximum observed ratio between the tangential forces and normal forces acting in each of the four contact points for each foot. These are referred to as the left and right friction ratio ($\mathbf{fr}_\text{l}$, $\mathbf{fr}_\text{r}$) indicators (both minimized),

$$\mathbf{fr} = \max\Big(\frac{\boldsymbol{ft}_{ci}}{\boldsymbol{fn}_{ci}}, \forall \boldsymbol{i} \in 1, 2, 3, 4\Big). \tag{5.2}$$

The function max() returns the maximum value of its inputs, $\boldsymbol{ft}_{ci}$ is the tangential component of the contact force at contact point $\boldsymbol{i}$, and $\boldsymbol{fn}_{ci}$ its normal component. These measurements refer to the maximums of the entire locomotion trial.

Experiment 1-C stability related locomotion features

For experiment 1-C, these two last features are removed, with two new being added: another indicator related to the friction ratio (global mean f.r. indicator), and the mean trajectory error of the ZMP reference of the controller.

The global f.r. indicator consists on the average observed ratio between the tangential forces and normal forces acting in each of the four contact points of both feet (as opposed to only one of them),

$$\mathbf{fr}_\text{global} = \frac{1}{8} \times \Big(\frac{\boldsymbol{ft}_{ci}}{\boldsymbol{fn}_{ci}}, \forall \boldsymbol{i} \in 1, 2, ...8\Big). \tag{5.3}$$

This average is applied to every feet sensors' measurement in the simulation.

The mean trajectory error is computed as follows,

$$\mathbf{e}_\text{mean} = \frac{h}{t_\text{f}} \times \sum_{t=0}^{t_\text{f}} \text{distance}(\text{trajectory}_t, \text{robot\_position}_t), \tag{5.4}$$

where distance(.) is the Euclidean distance between the three-dimensional points trajectory_t and robot_position_t, which are the robot's pretended location according to the locomotion trajectory and its actual location, at time t.

Overall setup

Adding these new features to experiments 1-B was done with the intent of bringing more stability to the explored behaviors, which in turn could result in more trials that completed the 20 seconds of simulation and possibly reach higher speeds, while not being discarded because the robot fell down. Further experimenting with this approach in 1-C was done by combining the two indicators from 1-B and adding another stability related measure (the average trajectory error).

Table 5.2 shows the parameters used in each of the experiments, as well as the range of values used for the NSGA-II optimization. These ranges were selected with, first, consideration of the physical limits of the robot with variables such as the step length, and then manual experimentation of multiple values for each parameter to determine at what point they would cause the locomotion to fail. The ranges were further adjusted in the 1-B and 1-C experiments with the data gathered from the previous ones, by eliminating the values that resulted in failures in every trial. Table 5.3 shows the features optimized for in the optimization in each experiment.

Table 5.3: Locomotion features involved in each optimization process of the first stage of the iCub experiments, along with the best value reached for them. This translates in the maximum value for the mean speed, and the minimum for the rest of the features. Empty cells signal that the respective feature was not an objective of optimization in that experiment.

	Best value		
Feature	1-A	1-B	1-C
$\boldsymbol{v}_{\text{mean}}$ (m/s)	0.2129	0.2997	0.2741
$\boldsymbol{\tau}_{\text{total}}$ ($\text{N}^2.\text{m}^2.\text{s}$)	5800	7440	6748
$\mathbf{fr}_{\text{l}}$	—	0.2554	—
$\mathbf{fr}_{\text{r}}$	—	0.2461	—
$\mathbf{fr}_{\text{global}}$	—	—	0.0048
trajectory $\mathbf{e}_{\text{mean}}$	—	—	0.002854

5.1.3 Stage 1 results

Figures 5.3, and 5.4 show the average, maximum, and minimum values of the performance indicators reached in each generation of the optimizations of the first stage, for the cases where the robot did not fall. Table 5.3 shows the best values for each indicator for all experiments.

Comparing results from the three different setups

Overall, the best values for the speed and torque indicators are reached around the 20 and 30 generations into the NSGA-II optimization. Each experiment shows a trade-off

in achieving the highest speed and the lowest torque output, with 1-B reaching the highest speed overall and the highest torque output, 1-C showing intermediate results, and 1-A the lowest speed and torque output.

Comparing the obtained speeds with literature

The only reference we found to a speed value reached in an iCub experiment in a comparable context was in Ibanez, Bidaud, and Padois (2014), where they use a reference speed of 0.20 m/s, while using the same LQP controller from Salini, Padois, and Bidaud (2011) used in this work, with the simulation running in the Arboris-python simulator (*Arboris-python*). Another, more recent, work (Hu et al., 2016) performed experiments using different classic control techniques (*e.g.* ZMP, position control) in a version of the iCub without arms and head. These reached a speed of 0.037 m/s in level ground.

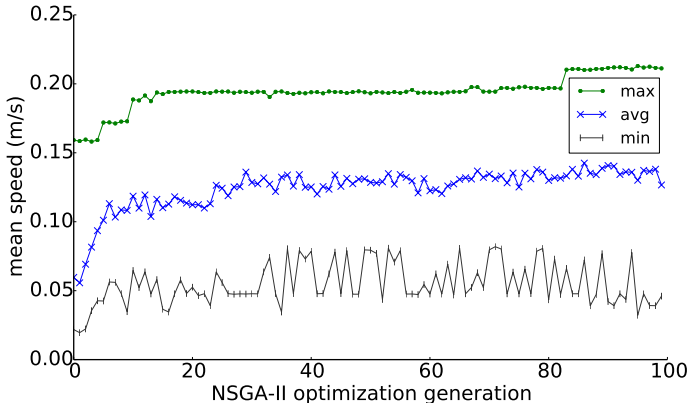
Effects of adding stability measures as optimization objectives

Adding the stability measures in 1-B and 1-C seems to have lead to more cases where the robot completed the trial without falling, allowing for behaviors that reach the full 20 seconds of the trials while still maintaining speeds that would cause it to fall in 1-A. This is supported by the higher number of failures registered in 1-A comparatively to 1-B and 1-C, as one can observe in Figure 5.5. A higher number of failures in a given generation is not necessarily unwanted, since exploring riskier behavior can lead to ones with higher speeds, while still being stable, but, taking into account the failure of the optimization of 1-A to obtain these behaviors, one can infer that this was not the case in that particular situation.

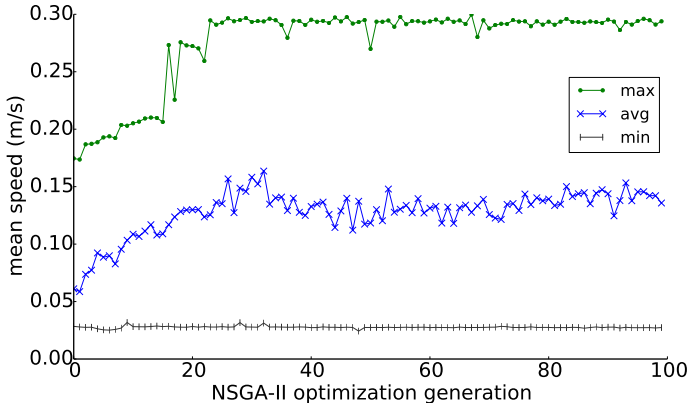
Regarding the fact that adding the stability measures in 1-B and 1-C resulted in worse results for the optimal values of the torque output indicator, one possibility is that this is caused by the focus on the stability measures, on top of the speed indicator. This can lead to more stable and/or fast behaviors that are not ideal to produce low torque outputs, which are instead traded-off for the other features.

Comparing the Pareto fronts of the three different setups

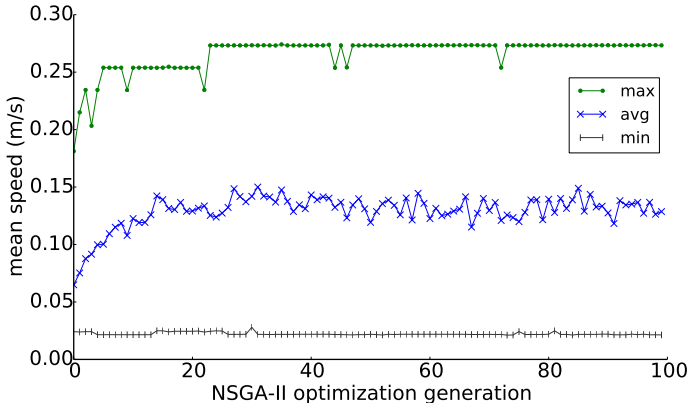
Figure 5.6 shows the Pareto fronts of the stage 1 optimizations, when only considering the mean speed and torque related indicators. These reinforce the notion that the three optimizations resulted in sets that show a trade-off between speed an torque output. 1-A shows low values of torque output not reached by the other sets, and 1-C shows high values of speed that 1-B does not reach, while also showing a higher concentration of solutions around the top side of the plot that is not present in either of the other sets. Although not optimal in either of the extremes, 1-B still shows a stretch of solutions that are optimal around the middle section of the conjunction of the Pareto fronts (highlighted in the figure). This set of solutions could still be useful if one was looking for the balance of mean speed and torque output that they provide, as oppose to focusing exclusively in one of the indicators.



(a)

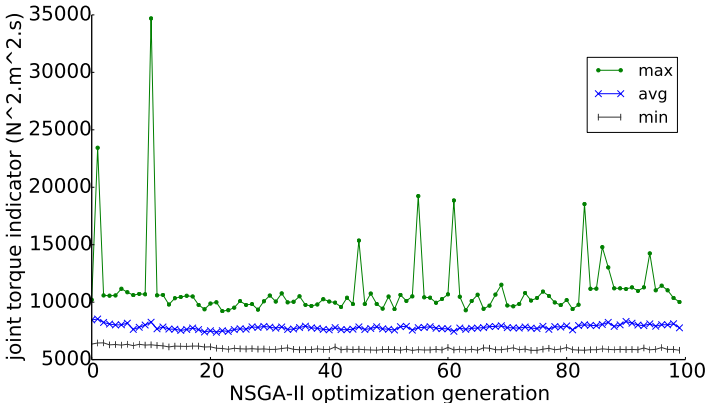


(b)

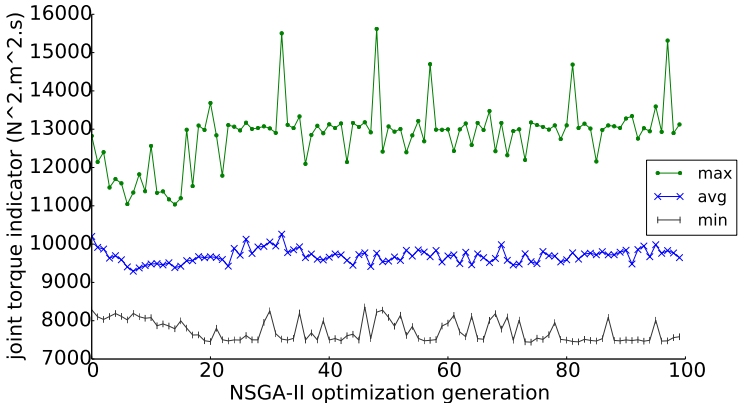


(c)

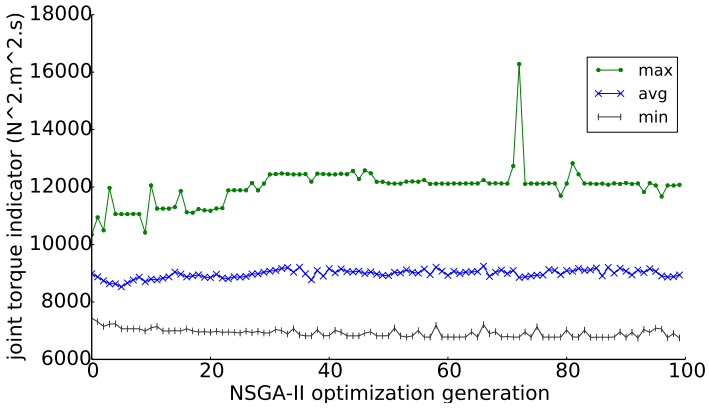
Figure 5.3: Maximum, average, and minimum speed for the simulations performed in each generation of the iCub optimization for stages 1-A (a), 1-B (b), and 1-C (c). Failed trials, where the robot fell, are not considered here.



(a)

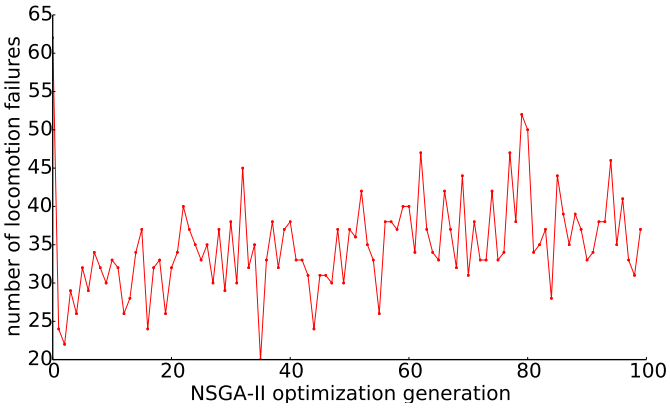


(b)

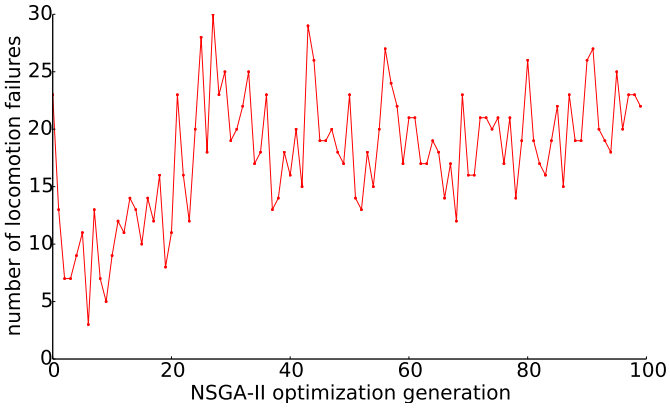


(c)

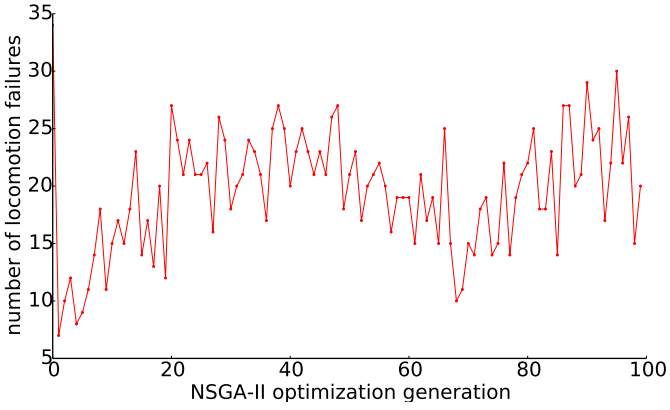
Figure 5.4: Maximum, average, and minimum values for the torque indicator for the simulations performed in each generation of the iCub optimization for stages 1-A (a), 1-B (b), and 1-C (c). Failed trials, where the robot fell, are not considered here.



(a)



(b)



(c)

Figure 5.5: Number of failed simulations in each generation of the iCub optimization for stages 1-A (a), 1-B (b), and 1-C (c). A trial is considered a failures if the robot fell before completing the 20 seconds of the simulation, at which point it was stopped.

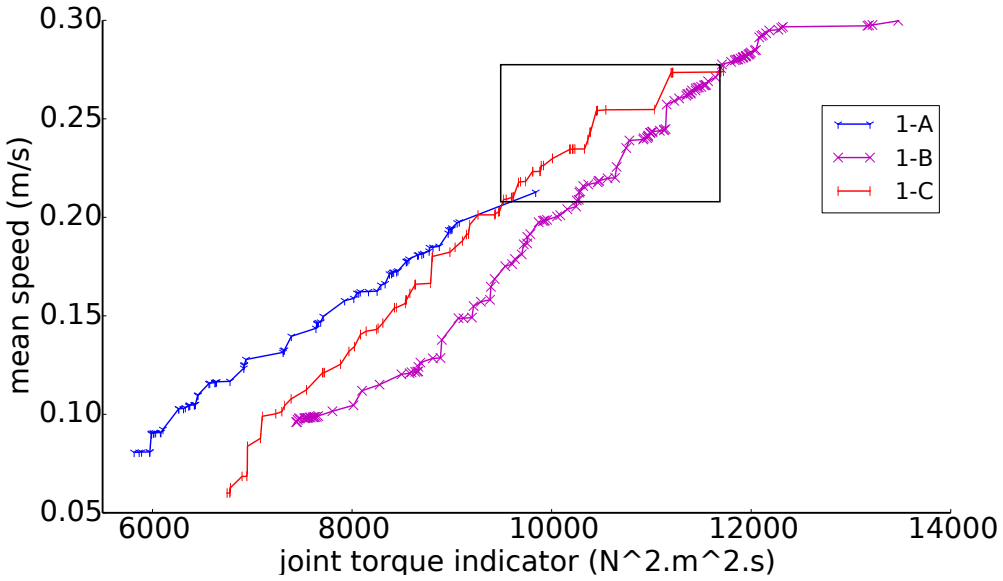


Figure 5.6: Pareto fronts of the stage 1 optimizations, relative to the mean speed and torque indicator objectives. These only consider successful trials, where the robot did not fall. A rectangle highlights an area where the Pareto front from 1-C has the optimal set of solutions.

5.1.4 Stage 2: Optimization on floors with different frictions

The setup for this optimization is very similar to the first one, with NSGA-II running a population of 100 individuals for 100 generations, and a forward walk for 20 s being the behavior optimized for. The penalties for the performance indicators in the case of the robot falling were maintained at 0 m/s for the mean speed and 99999 ($N^2.m^2$) for the torque output indicator. This setup was repeated for floors with different values of coefficient of friction (CoF), which is the value the physics simulation uses as the maximum value the ratio between the horizontal (tangential) and vertical (normal) forces acting on each of the feet's contact points can reach before the robot starts to fall,

$$\boldsymbol{ft}_c \leq \boldsymbol{fn}_c \times \text{CoF}. \tag{5.5}$$

This coefficient of friction is simulated with the Coulomb model of friction (see Appendix B) (Beer, Johnston, and Mazurek, 2013). The 10000 trials were repeated for different values of the floor's CoF: 0.05, 0.10, 0.25, 0.50, 0.75, 1.00, 1.25, and 1.50.

The parameters utilized were the same as described in Section 5.1.2, with the exception of the addition of the controller's restriction on the ratio between the normal and tangential components on the forces acting on the contact points between the feet and the ground, described in Appendix B. Higher values of this parameter result in an

Table 5.4: Range of values used for the parameters involved in the optimization process of the second stage of the iCub optimization.

Parameter	Parameter's range
Step length (m)	0.07 - 0.138
Step width (m)	0.05 - 0.10
Step height (m)	0.05 - 0.0395
Step duration (s)	0.5 - 3.0
Step phase ratio	0.525 - 0.94
ZMP horizon	1.025 - 2.0
Controller's CoF	0.05 - 2.0

Table 5.5: Locomotion features involved in the optimization process of the second stage of the iCub optimization.

Feature
$\boldsymbol{v}_{\text{mean}}$ (m/s)
$\boldsymbol{\tau}_{\text{total}}$ ($\text{N}^2.\text{m}^2.\text{s}$)
$\mathbf{fr}_{\text{global}}$
trajectory $\mathbf{e}_{\text{mean}}$

increased risk of the robot slipping and falling. This parameter will be referred to as the controller's coefficient of friction. The features optimized for were the same as in stage 1-C.

Table 5.4 shows the parameters used in the optimizations, as well as the range of values given to NSGA-II. The ranges used were those of past experiments from stage 1 (see Table 5.2), with the exception of the new parameter, the controller's CoF, whose range was selected in order to encompass the range of the floor's CoF the optimization were done in, with some margin given for the maximum value. Table 5.5 shows the features optimized for in the optimizations.

Sensitivity analysis setup

A sensitivity analysis was conducted with the results from the optimizations. The process is the one described in Section 4.4, with the solutions from the various Pareto fronts being tested for a set of different CoF values. The values tested were: 0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 1.25, and 1.5 (the same used in the previous optimizations).

5.1.5 Stage 2 results

Table 5.6 shows the best values included in the Pareto front of each of the optimizations from stage 2, as well as their size.

Impact of low values of CoF

Lower values of the CoF allow for smaller tangential forces on the contact points before the robot starts to slip, which translates in restricting the robot to safer behavior, otherwise it would fall and its fitness would be penalized. The only clear situation where this affected the best $\boldsymbol{v}_{\text{mean}}$ obtained was for a CoF of 0.05, where the best value is clearly lower than the rest.

Comparing optimal performances for different CoFs

Other than the 0.05 CoF case, every optimization seems to have reached comparable values, with the exceptions of 0.25 and 1.25 CoF, who had lower, and very close, optimal speeds. The most simple explanation for this occurrence is that these optimizations never had the mutations that allowed to reach these types of solutions. Both these optimizations may have been stuck in similar behaviors, seeing as their performance values are very close, although they belong to trials in different terrains. Tuning the exploration vs exploitation trade-off could help these situations, or it may be the case that this setup is the best overall, but these just occur as a consequence of NSGA-II's stochastic nature.

Effects of the CoF in the size of the Pareto fronts

The size of the Pareto fronts when filtered down for speeds above 0.23 m/s expectedly increases with higher values of CoF, with exceptions of 0.25, and 1.25, matching the lower maximum values of speed, and the proposed explanation. Interestingly, the values for the minimum ZMP trajectory error seem to generally increase with the CoF. This could be attributed either to these higher friction surfaces restraining the movement in a way that can prevent the robot from getting its center of pressure to the intended location in time, or it could simply be that the differences in these values is not significant enough to necessarily have a cause that greatly affects the behavior of the locomotion towards speed or stability. A similar case can be made for the values of minimum torque output, which do not seem to be affected by the CoF changes, and do not show a trend that matches with the outliers of 0.25 and 1.25.

As reference, 0.25 is a very low value of CoF, 0.47 is the CoF for aluminum and steel surfaces moving relative to each other, 0.42 to 0.62 for two steel surfaces, and 1.4 (a high, unlikely, value) for two aluminum surfaces (*Coefficients Of Friction*).

Overall, these optimizations seem to reach values similar to those of stage 1-C, which had a very similar setup, for a CoF of 1.5. The question that should now be posed is if these optimal behaviors for different CoF are reached with different solutions, or if there are solutions that result in optimal behaviors for every situation.

Sensitivity analysis

The success rates of each Pareto front of solutions in different terrains, differentiated by their value of floor CoF, can be seen in Table 5.7.

Table 5.6: Results for the optimizations of the iCubs locomotion control on floors with different frictions. The table shows the best values for the $\boldsymbol{v}_{\text{mean}}$, torque output, average f.r., and average trajectory error indicators for the Pareto front of solutions of those experiments, which are, for the $\boldsymbol{v}_{\text{mean}}$, the maximum values, and the minimum values for the other indicators. The table also shows the size of those Pareto fronts, and their size when the solution with $\boldsymbol{v}_{\text{mean}}$ under 0.23 m/s are removed.

	Best values					
Floor's CoF	$\boldsymbol{v}_{\text{mean}}$ (m/s)	τ_{total} ($N^2.m^2$)	Mean $\mathbf{fr}_{\text{global}}$	Trajectory $\mathbf{e}_{\text{mean}}$ (m)	P.f. size	P.f. size (speed > 0.23 m/s)
0.05	0.2384	6429	0.0075	0.0016	123	4
0.10	0.2732	7008	0.0047	0.0019	116	21
0.25	0.2543	6722	0.0048	0.0019	126	6
0.50	0.2735	6673	0.0049	0.0024	123	14
0.75	0.2736	6693	0.0052	0.0021	128	18
1.00	0.2732	6986	0.0050	0.0026	120	23
1.25	0.2541	6617	0.0049	0.0025	124	8
1.50	0.2735	6774	0.0049	0.0032	122	31

Trials in lower values CoFs as the cause for lower success rates

As expected, the Pareto fronts relative to a particular CoF showed no failed simulations when tested on terrains with that particular friction. Pareto fronts that resulted from optimizations in lower values of CoF show a higher average success rate across all floors. This is due to lower success rates for trials on floors with lower CoF than those they were optimized for, with most fronts having very low percentage of success for 0.05. These results are expected, since lower values of CoF lead to a constrain in the normal forces the robot can sustain in its contact points with the ground without falling.

Trials in higher values of CoFs tend to perform better

Pareto fronts optimized tend to perform better in floors with higher values of CoF than those they were optimized for, opposed to the ones with lower values of CoF. This trend is not universal and the differences in success rate are not very high.

Is there a solution capable of performing well in all cases?

One of the questions this analysis was meant to answer, is if there are sets of solutions that can perform ideally for every value of CoF tested, and therefore make the other sets useless. The front for 0.05 shows a success rate higher than 90 % for almost all of the CoF values, but this is still not ideal, and this question must be pursued further

Table 5.7: Success rates of the sensitivity analysis of the Pareto fronts of solutions for stage 2 of the iCub experiments. These are the ratios between the number of simulations where the robot did not fall, and the total number of simulations for that category. The last column shows the average of each row. The best success rate for each CoF tested, and the best average, are bolded.

Pareto front	CoF the Pareto front solutions were tested on, and respective success rates (%)								Average (%)
	0.05	0.1	0.25	0.5	0.75	1.0	1.25	1.5	
0.05	**100.00**	93.50	93.50	94.31	89.43	92.68	93.50	92.68	**93.70**
0.1	17.24	**100.00**	96.55	96.55	96.55	96.55	94.83	95.69	86.75
0.25	3.97	54.76	**100.00**	95.24	93.65	94.44	96.83	94.44	79.17
0.5	2.44	46.34	78.05	**100.00**	94.31	94.31	94.31	93.50	75.41
0.75	1.56	27.34	67.97	89.06	**100.00**	98.44	99.22	97.66	72.66
1.0	1.67	47.50	72.50	91.67	92.50	**100.00**	93.33	93.33	74.06
1.25	5.65	44.35	80.65	88.71	95.16	97.58	**100.00**	94.35	75.81
1.5	0.0	14.75	64.75	88.52	90.98	93.44	97.54	**100.00**	68.75

by inspecting whether these solutions also reach optimal values for the performance indicators.

Table 5.8 shows the number of successful solutions from each front in the different terrains. Given that the size of the fronts are similar, and the trend these results show is very similar to the previous table, it serves as an indication that the analysis of the success rates is not affected by certain Pareto fronts having a small number of solutions.

Comparison of the speed performances of the different fronts

Table 5.9 shows the best speed values each Pareto front achieved in the sensitivity analysis. Five out of the eight fronts show the best speed for the CoF they were optimized for, although in two of these cases they are not the only set with the best value. In general, the sets corresponding to higher CoF are better performing, achieving high mean speeds across the CoF values above the ones they were optimized for, with a threshold of about 0.2735 m/s above a CoF of 0.25.

Solutions achieve higher speeds in the floors they were optimized for

Similarly to the case of the success rates, the solutions for each set did not perform well when it came to values of CoF they were not optimized for. This is especially true in regard to the floor with 0.05 CoF, where other Pareto fronts have clearly worse results. There are two outliers to these trends, the fronts for CoF of 0.25 and 1.25, which were discussed in the previous section. These results bring the insight that the solutions of these two sets, not being optimal for their respective CoFs, also do not perform above that value in other floors. Conversely, other sets of solutions perform better than those for a CoF of 0.25 and 1.25, with mean speeds very close to the maximum observed.

Table 5.8: Number of successful solutions from each Pareto front for stage 2 of the iCub experiments. The last column shows the average of each row. The highest number of solutions for each CoF tested, and the best average, are bolded.

Pareto front	CoF the Pareto front solutions were tested on, and respective number of successful solutions								Average
	0.05	0.1	0.25	0.5	0.75	1.0	1.25	1.5	
0.05	**123**	115	115	116	110	114	115	114	**115.25**
0.1	20	**116**	112	112	112	112	110	111	100.625
0.25	5	69	**126**	120	118	119	122	119	99.75
0.5	3	57	96	**123**	116	116	116	115	92.75
0.75	2	35	87	114	**128**	**126**	**127**	**125**	93
1.0	2	57	87	110	111	120	112	112	88.875
1.25	7	55	100	110	118	121	124	117	94
1.5	0	18	79	108	111	114	119	122	83.875

This means that some of these behaviors can perform well in multiple floors, which is an interesting result for the adaptation phase of the framework.

Comparison of the torque performances of the different fronts

Table 5.10 shows the lowest values of total torque output each Pareto front achieved in the sensitivity analysis. Similarly to the mean speed results, five out of the eight pareto fronts have the best performance for the CoF they were optimized for. Unlike those results, there are not tied best values for a specific floor, and it also does not exist a clear bottleneck for performance, like the value 0.2735 m/s. The trend of solutions optimized for higher CoFs performing better than the rest for relatively high values remains here, as well as the relative poor performances for lower CoFs, especially for 0.05. Interestingly, the Pareto front for 1.25 has the best value in four cases. This matches the fact that it got the lowest total torque output result of all the optimizations (see Table 5.6), and is in juxtaposition with the fact it did not find at least one locomotion behavior that reached the high values of $\boldsymbol{v}_{\text{mean}}$ observed in other fronts.

5.1.6 Conclusions

These are the main conclusions taken from the experiments related to the iCub, that were drawn from the results discussed in sections 5.1.3 and 5.1.5.

- The optimization process was successful in improving, at least in part, the performance indicators of the locomotion.
  - The best speed values for this combination of robot and controller (around 0.2997 m/s) were higher than those found in literature (around 0.20 m/s).

Table 5.9: Highest speeds obtained in the sensitivity analysis of the Pareto fronts of solutions for stage 2 of the iCub experiments. The last column shows the average of each row. The best speed value for each CoF tested, and the best average, are bolded.

Pareto front	CoF the Pareto front solutions were tested on, and respective maximum speed obtained (m/s)								Average (m/s)
	0.05	0.1	0.25	0.5	0.75	1.0	1.25	1.5	
0.05	**0.2384**	0.2383	0.2382	0.2381	0.2383	0.2382	0.2381	0.2382	0.2382
0.1	0.1643	**0.2731**	**0.2729**	0.2729	0.2729	0.2729	0.2730	0.2729	0.2593
0.25	0.2011	0.2383	0.2543	0.2540	0.2541	0.2540	0.2541	0.2540	0.2455
0.5	0.1872	0.2674	0.2673	**0.2735**	0.2734	**0.2735**	0.2735	**0.2735**	**0.2612**
0.75	0.1272	0.2259	0.2619	**0.2735**	**0.2735**	**0.2735**	**0.2736**	**0.2735**	0.2478
1.0	0.1542	0.2575	0.2573	0.2731	0.2731	0.2731	0.2731	0.2731	0.2543
1.25	0.1746	0.2428	0.2542	0.2540	0.2541	0.2540	0.2542	0.2540	0.2427
1.5	—	0.1477	0.2577	**0.2735**	0.2734	**0.2735**	0.2734	**0.2735**	0.2216

Table 5.10: Lowest total torque output values obtained in the sensitivity analysis of the Pareto fronts of solutions for stage 2 of the iCub experiments. The last column shows the average of each row. The best value for each CoF tested, and the best average, are bolded.

Pareto front	CoF the Pareto front solutions were tested on, and respective minimum $\boldsymbol{\tau}_{\text{total}}$ obtained ($\text{N}^2.\text{m}^2$)								Average ($\text{N}^2.\text{m}^2$)
	0.05	0.1	0.25	0.5	0.75	1.0	1.25	1.5	
0.05	**6444**	7221	7476	7428	7388	7452	7452	7480	7293
0.1	7057	**7036**	7096	7141	7127	7122	7131	7149	7107
0.25	9207	7385	**6727**	6716	6897	6914	6667	6681	7149
0.5	9393	7639	7230	**6663**	6756	6870	6721	6724	7250
0.75	8689	7475	6786	6740	6699	6691	6701	6703	7060
1.0	9274	7178	7172	6962	7027	6994	6997	7024	7329
1.25	8218	7265	7123	6920	**6604**	**6623**	**6653**	**6629**	**7004**
1.5	—	7476	6835	6742	6662	6754	6789	6774	7178

- Changing the parameters optimized, and the optimization objectives, lead to trade-offs for the best values of the performance indicators.
- Using stability measures as optimization objectives can lead to a reduction in the number of failures, and an increase of the maximum speeds achieved, but also an increase in the minimum torque outputs.
- The number of generations chosen for the optimizations seems to be enough for the process to develop properly, and it could be reduced in order to save time.
- Changes in the floor friction can limit the performance values that can be achieved.
- The data supports that optimizations from stage 2 that did not reach expected performance values were affected by the stochastic nature of NSGA-II, rather than being limited by the floor's CoF.
- There is a need to optimize the locomotion control towards different environments/terrains.

5.2 Controller optimization of the DARwIn-OP on floors with different slopes

This section describes the experiments conducted with the DARwIn-OP model when controlled by a CPG system, for the exploration/training phase of the framework. These were optimizations of the locomotion control of the robot, achieved by tuning the parameters of the controller. These experiments were conducted in two stages. In the first stage a preliminary experiment was done in order to choose the parameters and features of the optimization out of a larger initial list. This allows to shorten the time spent on later optimizations, without negative impact in the resulting performance, making them more efficient. In the second stage, the optimization setup decided on which information gathered from the first stage was used to tune the locomotion behavior in floors with ramps with different slopes. A sensitivity analysis was conducted on the results of this last trials, in which the performance of solutions optimized for one slope was evaluated in other terrains, for which the solutions were not tuned.

5.2.1 Optimization framework setup

The experiments were conducted in a model of the DARwIn-OP robot, described in Appendix A, Section A.3. The controller used for the robot's description was a CPG based controller, described in Appendix C, with the simulations being ran in the Webots simulator, briefly described in Section 4.3.5, and in a context relative to the DARwIn-OP in Section A.3. The control parameters, that change the characteristics of motion primitives to affect the overall locomotion behaviors, were tuned towards multiple objectives by using the evolutionary computation C++ framework Sferesv2 (Mouret and Doncieux, 2010). Figure 5.7 shows the model of DARwIn-OP while walking up one

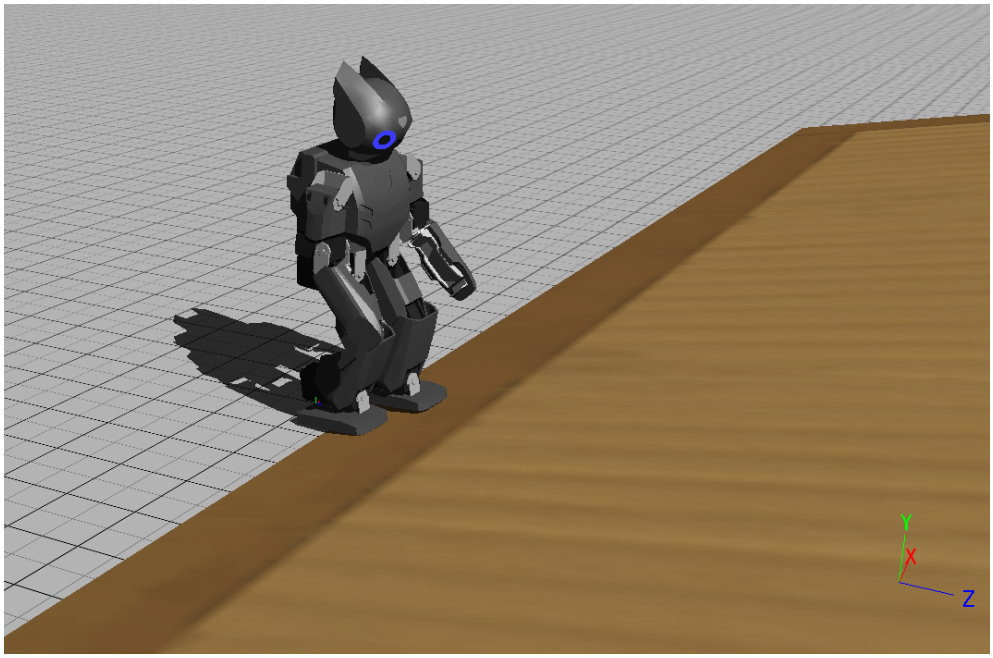


Figure 5.7: Screenshot of the DARwIn-OP walking up a ramp in a Webots simulation.

of the ramps tested for in the Webots simulations. Figure 5.8 shows a scheme of the optimization setup used in these experiments.

The NSGA-II parameters were the same ones used in the iCub experiments, shown in Table 5.1. Section 5.1.1 includes a brief discussion about how these variables affect the optimization process. The number of solutions in the population and the number of generations of the EA were the only values altered, and only in the first stage of the experiments.

A preliminary experiment was conducted, with the intent of analyzing the effect of the parameters and the locomotion features in the final performance indicators, and with that choose a setup for later optimizations based on fewer variables. This is what is referred to as stage 1 for these group of trials. In stage 2, the locomotion control was optimized for ramps with different slopes.

5.2.2 Stage 1: Preliminary optimization

The first optimization of the behavior provided by the CPG controller on the locomotion of the DARwIn-OP was done with the objective of tuning the parameters optimized and the objectives optimized for in such a way that the overall goals would be achieved.

The general goal of the whole optimization process is to obtain a repertoire of different behaviors, that will then be used to enable adaptation to different floor conditions. Since this adaptation is for a straight, forward walk, that is the motion that will be optimized for here. The duration of each trial is 20 seconds. The optimization is done by

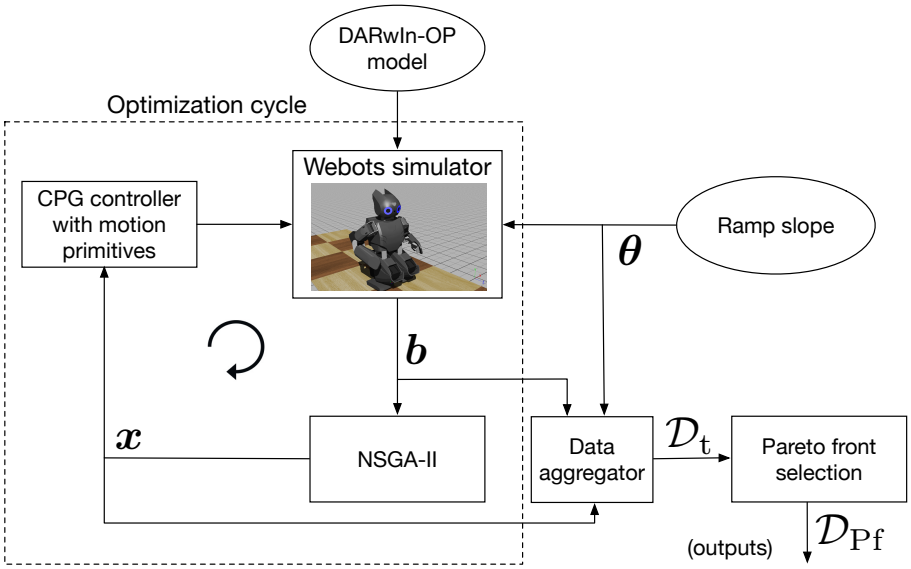


Figure 5.8: Architecture of the optimization process for the DARwIn-OP. $\boldsymbol{x}$ is the vector of values for the parameters being optimized, $\boldsymbol{b}$ is the vector of locomotion features results from a trial, $\boldsymbol{\theta}$ is the context variable, and $\mathcal{D}_t$ and $\mathcal{D}_{pf}$ are datasets containing all the data from the optimizations, and only the data related to the Pareto fronts of the optimizations, respectively. Refer to Figure 4.1 for a description of the framework.

NSGA-II (Deb et al., 2002), using its implementation in the evolutionary computation framework Sferesv2 (Mouret and Doncieux, 2010).

Parameters

The parameters used in this first optimization where the ones described in Section C.3.2 of Appendix C, and listed in Table 5.11. This leads to an optimization with twenty parameters, which would be too many, considering the time already spent in thousands of simulations in the iCub section for only six or seven parameters. For this reason, an optimization with a larger population and number of generations than the previous work was conducted, with a population of 200 solutions and 200 generations of evolution.

Locomotion features

Four optimization objectives were used: two performance related objectives, and two stability related ones, listed in Table 5.12. These are not the same ones used in the iCub experiments, for two reasons: 1) experimenting with different features can provide more information about which ones are more appropriate in different situations; 2) the stability measures were specific targeted at compensating for problems with the CPG controller.

Table 5.11: Reference indexes and minimum and maximum values for the parameters used in the preliminary optimization of the DARwIn-OP.

Parameter	Index	Minimum	Maximum
$\boldsymbol{\alpha}$	1	0.01	10
$\boldsymbol{T}(s)$	2	0.1	1.5
$\mathbf{A_{balancing,hip}}$	3	0	25
$\mathbf{A_{balancing,ankle}}$	4	0	22
$\mathbf{A_{flexion,hip}}$	5	0	80
$\mathbf{A_{flexion,knee}}$	6	0	44
$\mathbf{A_{flexion,ankle,h}}$	7	0	44
$\mathbf{A_{flexion,ankle,k}}$	8	0	44
$\mathbf{A_{compass,hip}}$	9	0	15
$\mathbf{A_{compass,ankle}}$	10	0	12
$\mathbf{A_{yield,knee}}$	11	0	17
$\mathbf{A_{yield,ankle}}$	12	0	5
$\mathbf{O_{hip,yaw}}$	13	-180	180
$\mathbf{O_{hip,pitch}}$	14	-40	5
$\mathbf{O_{knee,pitch}}$	15	30	55
$\mathbf{O_{ankle,pich}}$	16	13	27
$\mathbf{O_{hip,roll}}$	17	-67	10
$\mathbf{O_{ankle,roll}}$	18	-38	9
$\boldsymbol{\sigma}_{\mathbf{flexion,hip}}$	19	0.01	1.0
$\boldsymbol{\sigma}_{\mathbf{flexion,knee}}$	20	0.01	1.0

Table 5.12: Locomotion features involved in the preliminary optimization of the DARwIn-OP locomotion control.

Feature
$\boldsymbol{v}_{\text{mean}}$ (m/s)
$\boldsymbol{\tau}_{\text{mean}}$ ($\text{N}^2.\text{m}^2.\text{s}$)
Δ_{df}
CoP to support polygon distance (m)

Performance related features

The performance objectives used were the mean speed of the robot on the sagittal plane, $\boldsymbol{v}_{\text{mean}}$, and the mean sum of the squares of the torque values of all of the robot's joints, given by

$$\boldsymbol{\tau}_{\text{mean}} = \frac{h}{t_{\text{f}}} \times \sum_{t=0}^{t_{\text{f}}} \boldsymbol{\tau}_t \times \boldsymbol{\tau}_t. \tag{5.6}$$

where h is the duration of a time step in the simulation, t_{f} is the total duration of the simulation, and $\boldsymbol{\tau}_t$ is a vector containing the torque of every joint at simulation time t.

Stability related features

The stability objectives used were the difference between the duty factors of each foot, Δ_{df}, and the mean distance from the center of pressure (CoP) of the feet to the center of the support polygon of the robot.

The duty factor is defined as the ratio between the time the foot spends in contact with the floor and the total time of the locomotion task. The difference is then

$$\Delta_{\text{df}} = \text{abs}(\mathbf{df}_l - \mathbf{df}_r), \tag{5.7}$$

where abs() is a function returning the absolute value of the input, and $\mathbf{df}_l$ and $\mathbf{df}_r$ are the duty factors of each foot.

The CoP was calculated using the feedback from the four force sensors in each of the robot's feet. The support polygon is determined by in which of the three situations the robot is in: 1) the left foot is supporting the robot (the support polygon is the surface of the foot contacting the ground); 2) the right foot is supporting the robot; 3) both feet are supporting the robot (the support polygon is similar to the one observed in Figure 2.4 b)).

Overall setup

The speed related objective was maximized, while the other three were minimized. Simulations in which the robot fell, detected as the biped's waist going below a certain height, were stopped before the 20 s were over, and the mean speed $\boldsymbol{v}_{\text{mean}}$ was given a fitness penalty that scales with time,

$$\text{penalty} = 1 - \frac{t}{t_{\text{f}}}, \tag{5.8}$$

$$\boldsymbol{v}_{\text{mean}} = -1 \times \text{penalty} \times \boldsymbol{v}_{\text{penalty}}. \tag{5.9}$$

The penalty is the factor weighting how much of the $\boldsymbol{v}_{\text{penalty}}$ value will be used as fitness, going from 0 when the robot completes the 20 s of simulation, to 1 if the robot could fall immediately. The value of v_{penalty} used was 1.0 m/s.

Table 5.11 also shows the minimum and maximum values used in the optimization for each parameter. These values were determined by running simulations with default

parameter values taken from the work where the controller is described in Matos (2013); Matos and Santos (2014), and changing the value of a single parameter multiple times. The range of values of each parameter that did not cause the locomotion to fail (*i.e.* making the robot fall) were used in the EA optimization.

Correlation analysis

Using twenty parameters in the EA optimization requires a high number of generations with a large population. A smaller number of tuned parameters can lead to a more focused optimization, that results in better results in less time (Davidor, 1991).

In order to choose which parameters will be kept for the next optimization, a correlation analysis was conducted on the preliminary results. This procedure is described in Section 4.5.

5.2.3 Stage 1 results

The evolution of the performance related objectives results, for the simulations where the robot did not fall, throughout the 200 generations of the EA optimization, is shown on Figures 5.9 and 5.10.

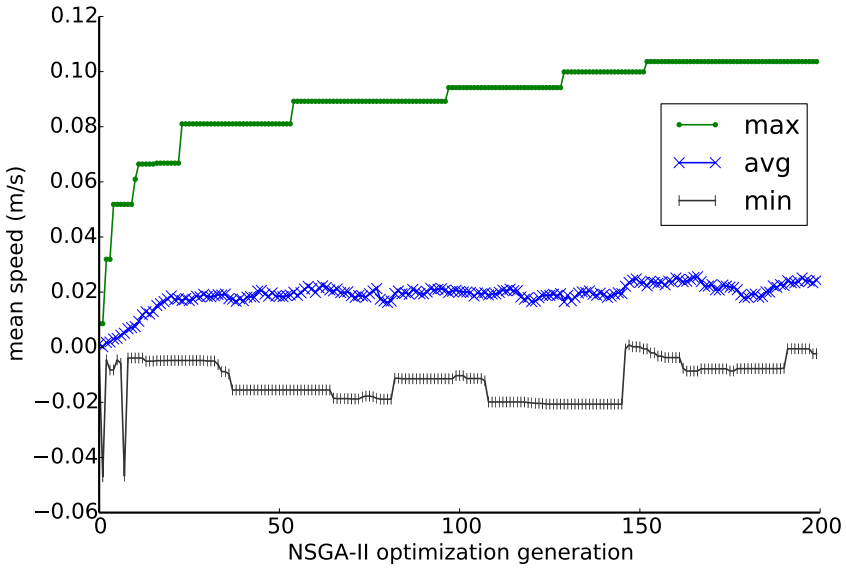


Figure 5.9: Maximum, average, and minimum values for the mean speed of the successful simulations throughout the 200 generations of the preliminary DARwIn-OP optimization.

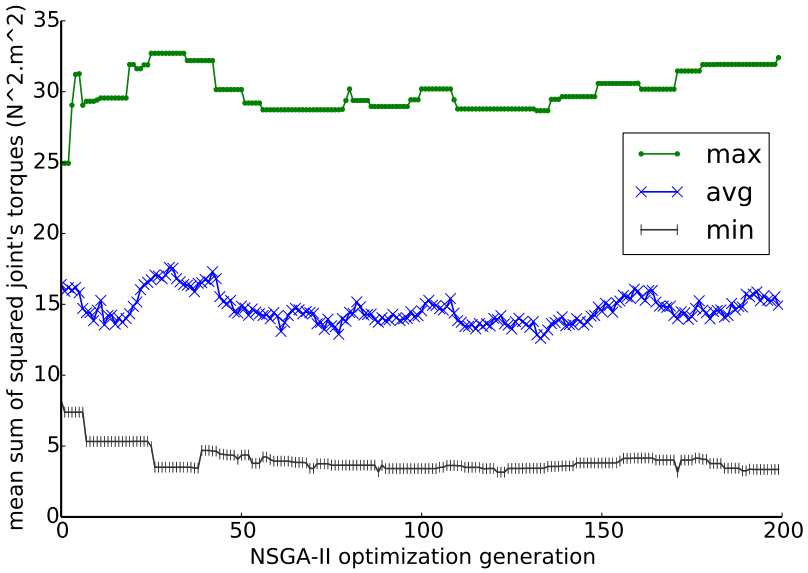


Figure 5.10: Maximum, average, and minimum values for the mean of the total torque output of the successful simulations throughout the 200 generations of the preliminary DARwIn-OP optimization.

Performance values achieved in the preliminary optimization

The optimization aimed at maximizing speed, and minimizing torque. Given this, a strictly increasing maximum speed can be observed, as well as a generally decreasing $\boldsymbol{\tau}_{\text{mean}}$ minimum. The biggest increases in performance occurred in the first 30 or so generations of the optimization, although smaller improvements can be seen in later generations.

How the way the CPG controller works can result in very low speeds

Negative values of mean speed are observed due to the fact that some solutions cause the robot to move backwards slightly. This happens with this controller, and not with the one used in the iCub optimizations, because it is based on motion primitives fed to each joint that can be modulated to produce such an effect. The combination of all the joint trajectories can lead to different types of robot orientation and general walking direction, and that is defined by the parameters of the controller. The dynamics based controller works in such a way that the trajectory set for each foot will be followed to at least a certain extent. This trajectory is defined at the beginning of the simulation, and is not affected by the parameters tuned.

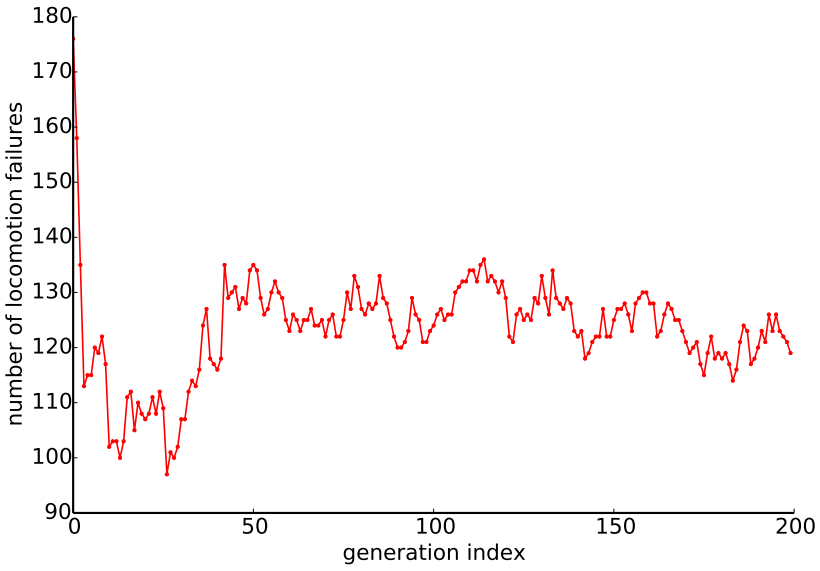


Figure 5.11: Number of failed locomotion attempts throughout the 200 generations of the preliminary DARwIn-OP optimization.

Comparing the obtained speeds with literature

To give context to the values of speed achieved, we have as comparison a couple of similar works achieving a speed of 0.10 m/s. Ha, Tamura, and Asama (2011) reached this number with a CPG based control of a DARwIn-OP model in simulation. Later, Oliveira et al. (2013) applied a multi-objective optimization with NSGA-II to the same CPG based controller used in the experiments with the DARwIn-OP (Matos and Santos, 2012), also in simulation, resulting in the same performance.

Discussing the number of failed simulations observed

Figure 5.11 shows the number of simulations where the locomotion failed in each generation. The number of failed simulations increasing does not necessarily mean the optimization is ineffective, since riskier behavior that can lead to higher performances can also lead to more failed attempts. As long as the number of failures is not excessively high, with around 130 out of the 200 being acceptable, that should not pose a problem.

Correlation analysis

Table 5.13 contains the results of the correlation analysis conducted by determining the measures detailed above. These results can be used to decide on which parameters (optimization inputs) and locomotion features (optimization objectives) to remove from

further stages, making the optimizations faster and more efficient. To this end, a direct comparison of each parameter's variation and correlation with each other is helpful. Table 5.14 shows a comparison of each value from Table 5.13 to its mean in the entire parameter population (the mean of each column).

Clarifying the importance of negative correlation values

Negative correlations are also desirable, since they only mean that the parameter should be decreased, and not increased, in order to obtain a better result. Because of this, only the absolute values were considered when calculating the mean and subtracting it to each cell. Positive values indicate higher variations or correlation than the population mean, with lower having the opposite indication. In this instance, negative values are undesirable, since they indicate the correlation is worse than the mean.

The correlation between parameters

Having a higher coefficient of variation means the optimization algorithm varied its value more frequently, indicating that it had incentive to do so, and is therefore desirable. The Pearson correlation coefficient in relation to other parameters should be as low as possible, since if two parameters have a very high correlation there is not much point to keep both of them in the optimization. On the other hand, having high correlation with the optimization objectives is important, particularly with the performance indicators ($\boldsymbol{v}_{\text{mean}}$ and $\boldsymbol{\tau}_{\text{mean}}$). A high correlation with the success measure is also desirable.

The choice of parameters for the next optimizations

Seven parameters were selected, shown in Table 5.15, from the initial twenty that showed the overall best results according to the guidelines set above. The choice is very clear in this situation, at least for six of those, since $\mathbf{A_{yield,knee}}$ shows only slightly above average correlation with the objectives and the success indicator, although the objectives where it performs best are the performance indicators. The process of decision can be automatized by assigning a weight for each of the columns of Table 5.14, and adding them into a single value.

5.2.4 Stage 2: Optimization on floors with different slopes

The CPG controller was optimized for floors that presented different slopes, via the addition of ramps. The setup for these optimizations is similar to the one for the preliminary optimization in Section 5.2.2. The differences are exposed ahead.

Parameters

The parameters tuned in these optimizations were the ones highlighted in the discussion of the correlation analysis from Section 5.2.2, and listed in Table 5.15. Similarly to what

Table 5.13: Results of the correlation analysis conducted on the data from the preliminary optimization of the DARwIn-OP. The indexes are the ones listed in Table 5.11. The table includes the coefficient of variation for each parameter, the average of the Pearson correlation coefficients between each parameter and the rest of them, the mean of the Pearson correlation coefficients between each parameter and the optimization objectives, and the correlation coefficient with the success state of the simulation (1 if the robot did not fall during the simulation, 0 otherwise). Parameters highlighted in further analysis are bolded.

Index	cv	Pearson correlation coefficients						
		Parameters average	Objectives average	$\boldsymbol{v}_{\text{mean}}$	$\boldsymbol{\tau}_{\text{mean}}$	Δ_{df}	CoP	Success
1	43.35%	0.2386	0.1443	0.2553	-0.0149	0.0922	-0.0936	0.2652
2	**42.70%**	**0.1211**	**0.3661**	**0.3757**	**-0.3661**	**0.2544**	**0.4520**	**0.3823**
3	55.38%	0.1853	0.0887	-0.1209	0.0774	-0.0941	-0.0271	-0.1242
4	66.63%	0.1525	0.1606	0.2575	-0.0534	0.1010	0.1379	0.2532
5	61.38%	0.1328	0.1222	0.1103	-0.1604	0.1210	0.1213	0.0979
6	32.95%	0.1569	0.1788	0.2800	-0.0983	0.0733	0.1608	0.2818
7	47.30%	0.1685	0.1212	0.1777	-0.1373	0.0861	0.0243	0.1805
8	**67.75%**	**0.1914**	**0.4059**	**0.4952**	**-0.2499**	**0.2824**	**0.5163**	**0.4856**
9	79.29%	0.1492	0.1764	0.2088	-0.1831	0.1399	0.1451	0.2050
10	56.39%	0.1191	0.0832	0.1569	-0.0002	0.0960	0.0162	0.1468
11	**55.63%**	**0.1810**	**0.2346**	**-0.3227**	**0.2321**	**-0.1841**	**-0.1060**	**-0.3283**
12	53.74%	0.0755	0.0449	0.0177	0.0909	0.0003	-0.0933	0.0221
13	**25.80%**	**0.1949**	**0.4815**	**-0.6519**	**0.3576**	**-0.3917**	**-0.3600**	**-0.6466**
14	**60.80%**	**0.2200**	**0.4327**	**-0.4389**	**0.4169**	**-0.2715**	**-0.6089**	**-0.4275**
15	19.58%	0.2169	0.0685	-0.0319	-0.1106	0.1026	0.0584	-0.0390
16	13.38%	0.1854	0.1324	-0.1893	0.0865	-0.0836	-0.1171	-0.1857
17	21.05%	0.1429	0.0704	0.0176	0.0728	-0.1493	-0.0911	0.0211
18	25.98%	0.1390	0.1455	0.1803	0.1990	-0.0972	-0.0702	0.1810
19	**38.06%**	**0.1647**	**0.3317**	**0.4013**	**-0.2421**	**0.2427**	**0.3680**	**0.4043**
20	**53.84%**	**0.2291**	**0.4826**	**0.5794**	**-0.4864**	**0.3306**	**0.4344**	**0.5823**

Table 5.14: Results from Table 5.13 with the mean of the respective column subtracted to each cell value. The mean was calculated with the absolute value of each column cell, so negative correlations were taken into account correctly. Each cell shows how higher or lower the value is for the mean of the entire population of parameters. Parameters selected for later optimizations are in bold.

		Pearson correlation coefficients differences						
Index	cv difference	Param. average	Obj. average	$\boldsymbol{v}_{\text{mean}}$	$\boldsymbol{\tau}_{\text{mean}}$	Δ_{df}	CoP	Success
1	-7.82%	0.0704	-0.0693	-0.0081	-0.1669	-0.0675	-0.1065	0.0022
2	**-8.47%**	**-0.0472**	**0.1525**	**0.1122**	**0.1843**	**0.0947**	**0.2519**	**0.1193**
3	4.22%	0.0171	-0.1249	-0.1425	-0.1044	-0.0656	-0.1730	-0.1388
4	15.46%	-0.0157	-0.0530	-0.0060	-0.1284	-0.0587	-0.0622	-0.0098
5	10.21%	-0.0355	-0.0914	-0.1532	-0.0214	-0.0387	-0.0788	-0.1652
6	-18.22%	-0.0113	-0.0348	0.0165	-0.0835	-0.0864	-0.0393	0.0188
7	-3.87%	0.0002	-0.0924	-0.0857	-0.0445	-0.0736	-0.1758	-0.0825
8	**16.58%**	**0.0232**	**0.1923**	**0.2318**	**0.0682**	**0.1227**	**0.3162**	**0.2226**
9	28.12%	-0.0190	-0.0373	-0.0547	0.0013	-0.0198	-0.0550	-0.0580
10	5.22%	-0.0491	-0.1304	-0.1065	-0.1816	-0.0637	-0.1839	-0.1162
11	**4.46%**	**0.0127**	**0.0210**	**0.0592**	**0.0503**	**0.0244**	**-0.0941**	**0.0653**
12	2.57%	-0.0927	-0.1687	-0.2458	-0.0908	-0.1594	-0.1068	-0.2409
13	**-25.37%**	**0.0266**	**0.2679**	**0.3885**	**0.1758**	**0.2320**	**0.1599**	**0.3835**
14	**9.63%**	**0.0517**	**0.2191**	**0.1754**	**0.2351**	**0.1118**	**0.4088**	**0.1645**
15	-31.59%	0.0486	-0.1451	-0.2315	-0.0712	-0.0571	-0.1417	-0.2240
16	-37.79%	0.0171	-0.0812	-0.0742	-0.0953	-0.0761	-0.0830	-0.0773
17	-30.12%	-0.0254	-0.1432	-0.2459	-0.1090	-0.0104	-0.1090	-0.2420
18	-25.19%	-0.0292	-0.0681	-0.0832	0.0172	-0.0625	-0.1299	-0.0821
19	**-13.11%**	**-0.0036**	**0.1181**	**0.1379**	**0.0603**	**0.0830**	**0.1679**	**0.1413**
20	**2.67%**	**0.0609**	**0.2690**	**0.3159**	**0.3046**	**0.1709**	**0.2343**	**0.3193**

Table 5.15: Reference indexes and minimum and maximum values for the parameters chosen after the correlation analysis of the DARwIn-OP. These were used for the stage 2 optimization.

Parameter	Index	Minimum	Maximum
$\boldsymbol{T}$	2	0.2	1.5
$\mathbf{A_{flexion,ankle,k}}$	8	23	44
$\mathbf{A_{yield,knee}}$	11	0	12
$\mathbf{O_{hip,yaw}}$	13	-180	0
$\mathbf{O_{hip,pitch}}$	14	-40	-20
$\boldsymbol{\sigma}_{\mathbf{flexion,hip}}$	19	0.01	1.0
$\boldsymbol{\sigma}_{\mathbf{flexion,knee}}$	20	0.01	1.0

Table 5.16: Locomotion features involved in the second stage optimizations of the DARwIn-OP locomotion control.

Feature
$\boldsymbol{v}_{\text{mean}}$ (m/s)
$\boldsymbol{\tau}_{\text{mean}}$ ($\text{N}^2.\text{m}^2.\text{s}$)
trajectory mse
CoP to support polygon distance (m)

was done in the first stage of the iCub optimizations, the ranges taken into account by NSGA-II for each parameter were tuned according to the results from the preliminary experiment, removing values that only resulted in failures.

Locomotion features

The performance related optimization objectives used were the mean speed of the robot on the sagittal plane [2] $\boldsymbol{v}_{mean}$ (maximized), and the torque indicator $\boldsymbol{\tau}_{\text{mean}}$, defined in Equation 5.6 (minimized).

For stability, the measures used as objectives were the mean squared error (MSE) of the locomotion trajectory (minimized), and the mean distance from the CoP of the feet to the center of the support polygon of the robot (minimized; see Section 5.2.2 for more details). The trajectory is defined as a straight line in the sagittal plane. This was done in order to further ensure the robot walks in the intended direction, since, in the preliminary optimization, even when considering the distance for the speed objective in the sagittal plane only, the robot did not walk in that direction for the highest speeds of the optimization. In relation to the preliminary optimization, the difference between the duty factor of each foot was removed for the addition of the MSE of the locomotion trajectory.

These optimization objectives are listed in Table 5.16.

Context variable values

This setup was applied to floors with no slope (no ramp), and for floors with slopes of ranging from 1 to 12 degrees, with a 1 degree intervals.

Sensitivity analysis setup

The solutions of the Pareto fronts of the optimizations were tested in a variety of different slopes, in order to analyze how their performance would differ in these different conditions. The process applied is described in Section 4.4. They were tested for the same values optimized for (from 0 to 12 degrees slopes, with a 1 degree interval).

[2] This plane is considered perpendicular to the ramps, in the optimizations these are present.

5.2.5 Stage 2 results

Table 5.17 shows some statistics regarding the stage 2 experiments. The results were restricted to values of $\boldsymbol{v}_{\text{mean}}$ above 0.0075 m/s, equivalent to walking at least 15 cm in the desired direction of movement, in 20 seconds of simulation. This was done so solutions where the robot barely moved, or did not move at all, were not considered towards lower torque indicator values or in the overall analysis of the size of the Pareto fronts.

Comparing the performances for optimizations in different ramps

There is an overall (expected) trend of lower maximum values of mean speed with increased values of slope, with a few low rises for the lower values of slopes, in addition to one for the 12 degrees front, in relation to both the 10 and 11 degrees results. The minimum values of $\boldsymbol{\tau}_{\text{mean}}$ follow the inverse trend, with generally higher values with increased values of floor slope. A significant outlier to this trend is the 10 degrees front, which has the worst value of the optimizations by a clear margin. There is a clear decline in the optimum mean speed above 8 and 9 degrees, and a clear incline for $\boldsymbol{\tau}_{\text{mean}}$ above 8 degrees. This difference is also observed in the size of the Pareto fronts, which predictably got smaller with increased slopes, with a few minor outliers.

Sensitivity analysis

Tables 5.18, 5.19, 5.20, and 5.21 show various statistics regarding the sensitivity analysis on the slope value of the ramp present in the simulation.

Why the iCub did not need a filter for minimum speed

The success rates from Table 5.18 deviate from the ones seen in the second stage of the iCub experiments, seen in Table 5.7, not because a speed minimum was imposed in this case, but because it was needed. With the iCub trials, the minimum of 0.0075 m/s is in fact observed for every trial meaning using it as a cutoff value would not change the results. With the DARwIn-OP simulations the 15 cm displacement may not be observed, mainly for two reasons. First, a ramp causes different restrictions than the ones in place because of changes in the value of floor friction. The ramp is an actual obstacle that can physically prevent a robot from walking forward, if not approached in a effective manner. The second reason is related to the control approach used for each robot. The dynamics based controller used in the iCub ensures that the robot follows pre-computed step positions to some degrees, and, since the task chosen is that of forward walking, there is a minimum speed it ensures, unless the robot falls, or encounters obstacles (which are not present in those trials). The CPG based controller, on the other hand, uses joint trajectories modulated by some of the parameters tuned in these optimizations, which can result in trajectories that do not cause displacement of the robot's center of mass, meaning it does move in the intended way, *i.e.*, resulting in forward locomotion.

Table 5.17: Results for the optimizations of DARwIn-OP's locomotion control on different slopes. The table shows the maximum/minimum (depending on whether the objective was maximized or minimized) values for the speed and torque indicators for the Pareto front of solutions of those experiments, filtered for solutions with a mean speed of at least 0.01m/s. The table also shows the size of those Pareto fronts.

	Pareto front filtered for mean speed > 0.0075 m/s		
Slope (degrees)	$\boldsymbol{v}_{\text{mean}}$ maximum (m/s)	$\boldsymbol{\tau}_{\text{mean}}$ minimum ($N^2.m^2$)	Size
0	0.1538	5.7900	97
1	0.1562	5.8903	88
2	0.1541	5.7391	81
3	0.1549	5.6856	89
4	0.1585	6.0199	69
5	0.1424	5.7429	62
6	0.1308	6.7458	41
7	0.1281	6.9924	46
8	0.1257	6.8873	22
9	0.0699	10.7721	21
10	0.0198	13.0866	6
11	0.0147	10.1730	7
12	0.0233	11.6505	5

Table 5.18: Success rates of the Pareto fronts for the sensitivity analysis in DARwIn-OP's stage 2 experiments. Each column shows the success rate for each Pareto front on a specific slope. Only successful simulation with a mean speed over 0.0075m/s were considered. The last column shows the average of all the values in that row. Best values for each slope tested, and the best total, are bolded.

Pareto front (degrees)	Slope the Pareto front was tested on (degrees), and respective success rate (%)													Average (%)
	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	90.4	76.0	76.0	75.0	45.2	34.6	25.0	11.5	6.7	4.8	0.0	1.9	0.0	34.4
1	78.8	84.6	70.2	59.6	44.2	24.0	25.0	19.2	11.5	6.7	1.0	0.0	4.8	33.1
2	94.3	86.2	**94.3**	87.4	81.6	55.2	36.8	27.6	18.4	10.3	3.4	2.3	4.6	46.3
3	**95.8**	83.2	92.6	**95.8**	80.0	62.1	45.3	30.5	20.0	21.1	**16.8**	**12.6**	2.1	**50.6**
4	84.1	**90.2**	86.6	86.6	**82.9**	69.5	51.2	28.0	9.8	11.0	7.3	4.9	2.4	47.3
5	86.7	81.3	78.7	84.0	82.7	**86.7**	**62.7**	45.3	**26.7**	4.0	5.3	2.7	0.0	49.7
6	73.4	68.4	72.2	67.1	59.5	62.0	49.4	26.6	15.2	3.8	2.5	0.0	0.0	38.5
7	85.5	86.8	85.5	75.0	72.4	64.5	59.2	**56.6**	18.4	13.2	6.6	3.9	3.9	48.6
8	78.8	75.3	64.7	57.6	47.1	49.4	27.1	12.9	25.9	9.4	5.9	2.4	1.2	35.2
9	53.7	47.6	42.7	41.5	31.7	37.8	24.4	20.7	24.4	**24.4**	7.3	2.4	0.0	27.6
10	23.2	23.2	17.1	15.9	13.4	23.2	8.5	8.5	4.9	6.1	7.3	0.0	0.0	11.6
11	52.7	37.4	41.8	31.9	38.5	29.7	26.4	15.4	15.4	12.1	7.7	6.6	0.0	24.3
12	63.4	45.1	35.4	20.7	14.6	11.0	9.8	6.1	3.7	4.9	4.9	2.4	**6.1**	17.5

Table 5.19: Size of the Pareto fronts for the sensitivity analysis in DARwIn-OP's stage 2 experiments. Each column shows the number of simulations with an average speed of at least 0.0075m/s for each Pareto front on a specific slope. The last column shows the total of all the values in that row. Best values for each slope tested, and the best total, are bolded.

Pareto front (degrees)	Slope the Pareto front was tested on (degrees), and respective number of simulations with an average speed > 0.0075 m/s													
	0	1	2	3	4	5	6	7	8	9	10	11	12	Total
0	**94**	79	79	78	47	36	26	12	7	5	0	2	0	465
1	82	**88**	73	62	46	25	26	20	12	7	1	0	**5**	447
2	82	75	82	76	71	48	32	24	16	9	3	2	4	524
3	91	79	**88**	**91**	**76**	59	43	29	19	**20**	**16**	**12**	2	**625**
4	69	74	71	71	68	57	42	23	8	9	6	4	2	504
5	65	61	59	63	62	**65**	**47**	34	20	3	4	2	0	485
6	58	54	57	53	47	49	39	21	12	3	2	0	0	395
7	65	66	65	57	55	49	45	**43**	14	10	5	3	3	480
8	67	64	55	49	40	42	23	11	**22**	8	5	2	1	389
9	44	39	35	34	26	31	20	17	20	**20**	6	2	0	294
10	19	19	14	13	11	19	7	7	4	5	6	0	0	124
11	48	34	38	29	35	27	24	14	14	11	7	6	0	287
12	52	37	29	17	12	9	8	5	3	4	4	2	**5**	187

Table 5.20: Best speed values for the sensitivity analysis in DARwIn-OP's stage 2 experiments. Each column shows the maximum mean speed (m/s) of each Pareto front on a specific slope. The last column shows the average maximum mean speed for each Pareto front (cells with no value are considered as 0 (m/s). The best values for each slope tested, and the averages, are bolded.

Pareto front (degrees)	Slope Pareto front was tested on (degrees), and respective maximum mean speed observed (m/s)													Average (m/s)
	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	0.154	0.150	0.147	0.143	0.114	0.094	0.066	0.031	0.022	0.018	-	0.008	-	0.073
1	0.135	0.156	0.135	0.132	0.093	0.062	0.095	0.087	0.029	0.019	0.013	-	0.011	0.074
2	0.139	0.141	0.154	0.144	0.141	0.134	0.098	0.077	0.055	0.011	0.008	0.008	0.009	0.086
3	0.149	0.150	0.151	0.155	0.140	0.135	**0.134**	0.120	0.043	0.030	0.026	0.022	0.008	**0.097**
4	**0.158**	**0.160**	**0.155**	**0.158**	**0.159**	0.105	0.117	0.103	0.043	0.022	0.021	**0.027**	0.009	0.095
5	0.136	0.138	0.133	0.142	0.141	**0.142**	0.131	0.098	0.075	0.020	0.019	0.015	-	0.092
6	0.136	0.131	0.134	0.139	0.130	0.125	0.131	0.124	0.112	0.032	0.023	-	-	0.094
7	0.129	0.129	0.115	0.114	0.124	0.127	0.126	**0.128**	0.053	0.022	0.018	0.012	0.012	0.085
8	0.116	0.120	0.119	0.118	0.126	0.128	0.125	0.054	**0.126**	0.035	0.016	0.008	0.009	0.084
9	0.097	0.073	0.048	0.100	0.036	0.046	0.029	0.022	0.023	**0.070**	**0.029**	0.011	-	0.045
10	0.032	0.039	0.027	0.026	0.019	0.025	0.024	0.021	0.022	0.022	0.020	-	-	0.021
11	0.111	0.082	0.109	0.102	0.108	0.077	0.080	0.037	0.026	0.014	0.011	0.015	-	0.059
12	0.095	0.086	0.076	0.069	0.019	0.013	0.011	0.011	0.011	0.011	0.010	0.008	**0.023**	0.034

Table 5.21: Best $\boldsymbol{\tau}_{\text{mean}}$ values for the analysis in DARwIn-OP's stage 2 experiments. Each column shows the minimum torque indicator value ($N^2.m^2$) of each Pareto front on a specific slope. The last column shows the average minimum of the torque indicator for each Pareto front (cells with no value are considered as the maximum value of the rest of the cells). The best values for each slope tested, and the averages, are bolded.

Pareto front (degrees)	Slope pareto front was tested on (degrees), and respective minimum value of torque indicator ($N^2.m^2$)													Average ($N^2.m^2$)
	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	**5.79**	6.39	6.50	6.66	7.82	7.31	8.24	8.29	13.46	12.83	-	12.36	-	9.86
1	6.05	**5.89**	6.09	6.32	6.38	10.51	7.63	9.57	10.38	12.71	13.65	-	11.71	9.45
2	6.10	6.18	**5.74**	5.90	6.33	6.38	6.77	7.51	12.54	12.38	12.98	12.70	11.94	8.73
3	6.06	5.93	5.88	**5.69**	6.24	6.24	6.69	8.66	13.49	11.75	12.42	12.42	**11.03**	**8.69**
4	6.30	6.36	6.60	6.69	6.02	7.48	7.81	**6.46**	13.62	12.83	12.05	11.43	13.34	9.00
5	5.82	6.23	6.09	6.45	**6.00**	**5.74**	**6.24**	6.88	8.38	12.87	13.28	12.43	-	8.74
6	6.68	6.24	6.26	6.62	6.77	6.65	6.75	7.77	9.35	**7.92**	**11.52**	-	-	8.99
7	6.63	6.54	6.55	6.26	6.88	6.95	6.97	6.99	10.65	12.94	12.24	12.01	13.01	8.82
8	6.96	7.12	7.28	7.34	6.94	7.01	8.38	8.13	**6.89**	12.92	11.92	11.88	13.98	8.98
9	8.44	8.56	8.35	7.61	10.09	7.30	13.52	13.28	11.75	10.77	11.78	13.89	-	10.96
10	10.92	10.99	10.77	10.78	11.18	11.59	13.18	13.75	13.56	13.94	13.09	-	-	12.93
11	6.62	7.17	7.82	7.37	7.35	7.74	8.10	12.70	12.79	12.72	11.79	**10.17**	-	9.96
12	8.12	7.52	9.26	9.31	9.15	9.06	9.06	11.77	15.97	13.83	17.16	17.07	11.65	11.46

Fronts optimized for lower values of slope had better success rates

Table 5.18 shows the success rate of each Pareto front of solutions in the slopes tested. In this case, only solutions which produced a behavior with mean speeds above 0.0075 m/s were considered. Overall, the fronts optimized for a mid-range of values, around 2 to 7 degrees, seem to have better consistency. Behaviors tuned for more steep ramps will be, in general, less risky, and therefore have higher chances of success for ramps with lower values of slope. On the other hand, these behaviors may also be overtuned for ramps that approach a leveled floor. The fact that fronts optimized for the mid-range were more successful overall gives credence to the hypothesis that there is a relationship between these solutions, and respective behaviors, that is somewhat close to a linear one, making the solutions optimized for values closer to both extremes more successful.

How the size of the Pareto fronts support the previous point

To continue analyzing the hypothesis that a mid-range of values tends to perform better, while taking into account a broader number of performance indexes, one can look at Table 5.19. This shows, for each Pareto front, the number of solutions with a high enough speed and diverse enough $\boldsymbol{\tau}_{\text{mean}}$ to be considered an optimal solution in the Pareto sense. These show that fronts related to lower values of slope tend to have a larger number of interesting solutions. There is also an increase at the lower values, making the optimum range between 2 and 7 degrees. This translates in a mid-range optima, with a slant to the lower values. Lower values of slope tend to result in less risky behaviors, since these have a higher tendency to result from the robot trying to walk up a steeper ramp. This can lead to these solutions optimized for higher slopes being more applicable to lower values of slope, and give them an edge in relation to those fronts. The criteria to choose the Pareto front solutions is not strictly defined, since the advantages of having one or multiple solutions that result in these acceptable values of speed are not always clear. However, the differences these results show are clear enough to reach a general conclusion.

Discussing the performance values from the sensitivity analysis

The previous results can have better context when analyzing the best performance values for each front, as opposed to a number of solutions above a certain threshold. Table 5.20 shows the best speed values reached by every set of solutions. Most of the fronts would have the best behavior for the slope value they were optimized for, if it was not for the performances shown by the front for 4 degrees. This group of solutions shows the best values for the lower values of slope. This front also had the overall best speed in the original optimizations, as seen in Table 5.17. As for the average values, they follow a very similar trend to that of Table 5.19: the best averages are generally towards the fronts for the low to mid-range of slopes tested.

Table 5.21 shows the best values for the other performance indicator, the $\boldsymbol{\tau}_{\text{mean}}$. Here the trend follows more or less the already seen better values for solutions optimized

for slopes in the mid-range of values, with the best results coming from fronts between 2 and 8 degrees. A clear outlier lies at the front for 10 degrees, that already had a higher minimum than the neighborhood solutions, as seen in Table 5.17. In short, these results only reinforce the ideas provided by the $\boldsymbol{v}_{\text{mean}}$ results.

5.2.6 Conclusions

These are the main conclusions taken from the experiments related to the DARwIn-OP, supported by the results, and respective discussions, seen in sections 5.2.2 and 5.2.5.

- The performance of the robot's locomotion was, at least in part, improved by the optimization process.
  - The best speeds observed in the stage 2 optimizations (around 0.1538 m/s), are better than the ones found in literature for the DARwIn-OP using a CPG based controller (around 0.1 m/s).
- Different locomotion controllers may require different optimization setups, adding restrictions that assure the desired behavior (e.g., the DARwIn-OP not moving past the ramp, and failing to achieve an acceptable minimum speed).
- Different terrain parameters can also cause the need for these restrictions (the ramp may be a bigger restriction to achieve an acceptable minimum speed than the controller).
- The correlation analysis allowed to tune an initially large optimization setup into one that is much less time consuming and computationally expensive, while improving the values of the main performance indicator.
  - The number of parameters in the optimization setup went from 20 to 7, the number of optimization objectives from 5 to 4, the number of generations from 200 to 100, and the population size from 200 to 100.
  - The maximum speed achieved in stage 1 was below 0.11 m/s, much lower than the value observed in stage 2 for the same floor (0.1538 m/s).
- There is a trend of decreased performance with an increase of the floor slope. The maximum speeds trend lower, and the torque output higher.
- An optimization from stage 2 (for the 10 degrees ramp) that did not reach expected performance values seems to have been affected by the stochastic nature of NSGA-II, rather than being limited by the ramp's slope.
- Fronts related to lower values of slope tend to have a larger number of solutions, and better average performances.
  - The Pareto fronts with more solutions were between the 2 and 7 degrees range, while the best overall performing front was for the 4 degrees optimization.

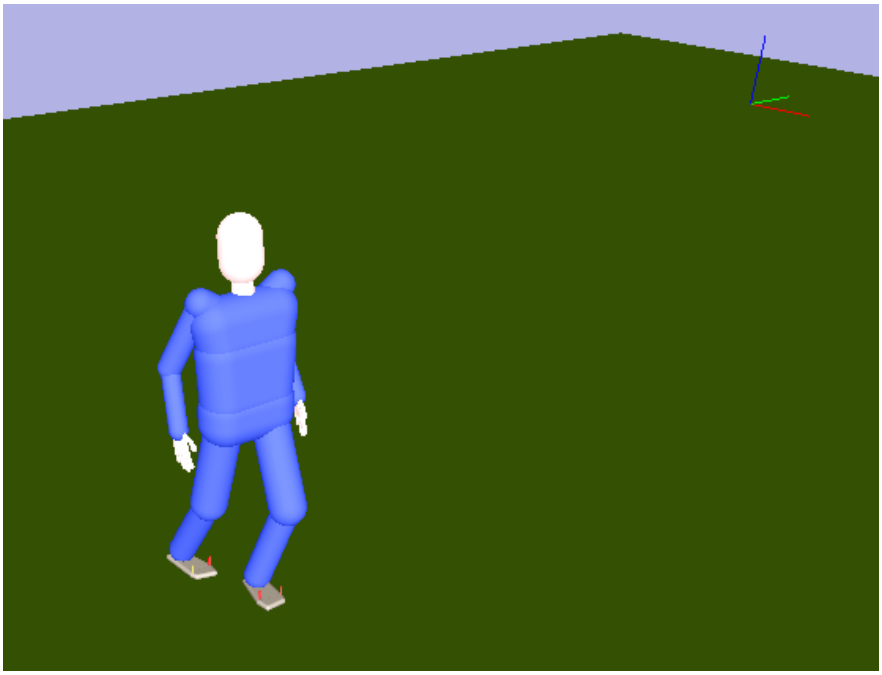


Figure 5.12: Screenshot of the XDE-manikin during simulation.

5.3 Effects of mass and volume changes on the control of a virtual humanoid robot

During locomotion, on top of changes to the environment surrounding a humanoid robot, changes to the robot itself can also affect the overall behavior on the task at hand. In this section, we explore the impact of mass and volume changes to the locomotion control of the XDE-manikin. Changing the mass of this system makes the weight of its body parts change proportionally, and the same happens with the height and volume of these parts, keeping the kinematic structure proportional. The locomotion control of the manikin, under the same dynamics based controller used in the iCub experiments, is optimized for different configurations of height and mass of the robot.

5.3.1 Optimization framework setup

The model of the XDE-manikin is described in Appendix A, Section A.2. The controller is the same one used in the iCub optimizations, a dynamics based controller from Salini described in Appendix B, and the simulations are conducted in XDE (described in Section 4.3.5). The optimization algorithm is NSGA-II, described in Section 4.3.6. Figure 5.12 shows the model of the manikin in the XDE simulations, while Figure 5.13 shows a scheme of the optimization setup.

The NSGA-II related parameters used in this experiment are the same as the ones

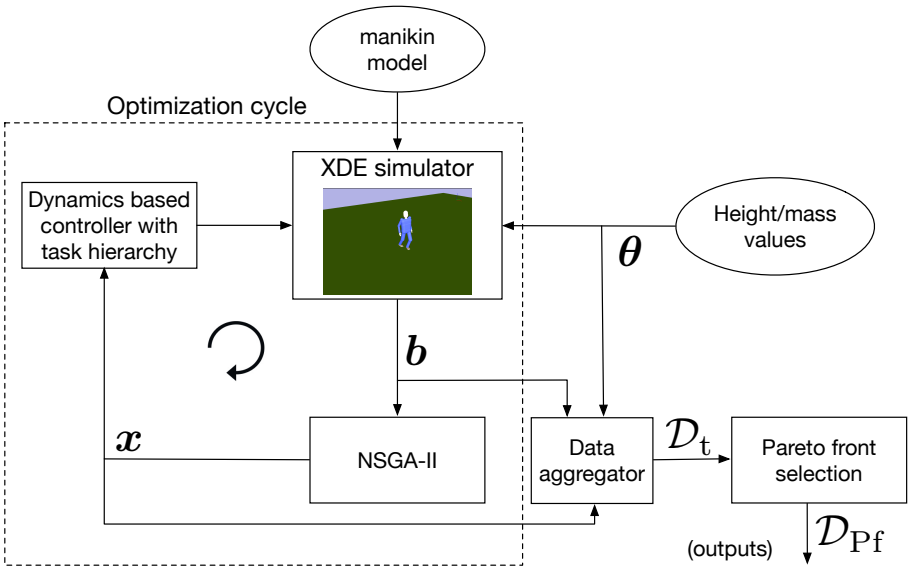


Figure 5.13: Architecture of the optimization process for the XDE-manikin. $\boldsymbol{x}$ is the vector of values for the parameters being optimized, $\boldsymbol{b}$ is the vector of locomotion features results from a trial, $\boldsymbol{\theta}$ is the context variable, and $\mathcal{D}_{\mathrm{t}}$ and $\mathcal{D}_{\mathrm{pf}}$ are datasets containing all the data from the optimizations, and only the data related to the Pareto fronts of the optimizations, respectively. Refer to Figure 4.1 for a description of the framework.

Table 5.22: Range of values used for the parameters involved in each optimization process of the XDE-manikin optimization.

Parameter	Range
Step length (m)	0.03 - 0.45
Step width (m)	0.075 - 0.175
Step height (m)	0.001 - 0.25
Step duration (s)	0.25 - 3.0
Step phase ratio	0.4 - 1.0
Controller's CoF	0.005 - 10.0

Table 5.23: Locomotion features involved in the XDE-manikin optimizations.

Feature
$\boldsymbol{v}_{\text{mean}}$ (m/s)
$\boldsymbol{\tau}_{\text{total}}$ ($\text{N}^2.\text{m}^2.\text{s}$)
$\mathbf{fr}_{\text{global}}$
trajectory $\mathbf{e}_{\text{mean}}$

in the iCub setup, shown in Table 5.1. Section 5.1.1 contains a discussion on these parameters.

5.3.2 Locomotion control optimization under different body's mass and height values

The locomotion control optimizations are applied to a task consisting of walking forward for 20 s. NSGA-II ran for 100 generations, with a population size of 100, resulting in 10000 trials. Runs which ended with a physics or/and controller failure, resulting in a robot fall and/or explosion, were given fitness penalties of 0 m/s for the mean speed, and 99999 ($\text{N}^2.\text{m}^2$) for the joint torque output indicator.

The parameters involved in the optimization are seen in Table 5.22. The objectives are listed in Table 5.23, and both the parameters and objectives used here are described in Section 5.1.2. The parameter ranges were selected in a way that allows for the optimization algorithm to find good values for any set of height and mass used. This is done by using the trial and error process described in Sections 5.1.2, and 5.2.2 for both the smallest and lightest robot and the tallest and heaviest one.

The 10000 simulations are repeated for different values of the robot's height and mass. Three height values are used: 1.60 m, 1.75 m, 1.90 m. For each height value, three different body mass indexes (BMIs) were used: 20, 25, 30. The BMIs were calculated using the equation $\text{BMI} = \frac{\text{mass}}{\text{height}^2}$, resulting in the combinations of height and mass values shown in Table 5.24, labeled as different sets in order to make displaying and discussing the experiment's easier.

Table 5.24: Sets of height and mass used in the XDE-manikin experiments, labeled as different sets for future reference.

Set	Height (m)	Mass (kg)
1	1.60	51.2
2	1.60	64
3	1.60	76.8
4	1.75	61.25
5	1.75	76.5625
6	1.75	91.875
7	1.90	72.2
8	1.90	90.25
9	1.90	108.3

5.3.3 Results

Table 5.25 shows some statistics regarding the experiments of different weighted and sized XDE-manikin models.

How the robot's size affects the maximum speed achieved

The results for the highest mean speed show a clear separation between the three different heights tested, with the highest speed from a robot with given height being still lower than the lowest speed of a taller one. This is not surprising, since taller robots have higher step lengths, and the step duration can be kept intact, albeit at the cost of a higher torque output.

How the robot's weight affects the maximum speed achieved

Heavier robots reached higher speeds, in general, in a given height category. There is an exception, for the height of 1.60 m, between sets 2 and 3. These values are very close though, with the same happening for the sets 8 and 9. One possible justification for these higher speeds is a higher momentum achieved by the robot's limbs when rotation around each other. This implies that higher forces are applied to those limbs, which is corroborated by the fact that the best speeds associated with higher masses are also associated with higher torque outputs.

The torque outputs observed in the different sets

The lowest torque outputs are also higher for taller robots, although there is not a clear separation like in the mean speed case. Within each height category, heavier robots generally present worse results, with an expectation in the transition from set 2 to set 3. Interestingly, this behavior was also observed for the mean speed optimum values, indicating that the optimization for set 2 may have focused its exploration in

Table 5.25: Results for the optimizations of the locomotion control of versions of the XDE-manikin with different height and total mass values. The table shows the best values achieved for the optimization objectives in those experiments, which are, for $\boldsymbol{v}_{\text{mean}}$, the maximum values, and the minimum values for the other indicators. The $\boldsymbol{\tau}_{\text{total}}$ for the behavior that showed the maximum speed for each set is shown in the next column. Finally, the table also shows the size of the Pareto fronts of the speed and torque objectives, and their size when the solution with a mean speed under 0.3 m/s are removed. The set number refer to the combinations of height and mass from Table 5.24

	Best values						
Set	$\boldsymbol{v}_{\text{mean}}$ (m/s)	$\boldsymbol{\tau}_{\text{total}}$ ($N^2.m^2$)	$\mathbf{fr}_{\text{global}}$	Trajectory $\mathbf{e}_{\text{mean}}$ (m)	$\boldsymbol{\tau}_{\text{total}}$ for the best speed ($N^2.m^2$)	P.f. size	P.f. size (speed $>$ 0.3 m/s)
1	0.3090	64667	0.0107	0.0045	215682	70	2
2	0.3562	122822	0.0105	0.0048	324877	68	14
3	0.3531	97000	0.0245	0.0106	496959	52	5
4	0.4262	118335	0.0107	0.0031	348012	85	25
5	0.4673	175234	0.0099	0.0031	519569	91	23
6	0.5060	252422	0.0094	0.0029	788532	72	26
7	0.5176	186186	0.0097	0.0014	578863	80	40
8	0.5905	299179	0.0096	0.0021	1039251	148	74
9	0.5955	411235	0.0091	0.0018	1257501	125	76

less efficient and faster behaviors, or, on the other hand, the optimization from set 3 may have been narrowed to efficient, but slower behaviors.

Discussing why heavier robot models could achieve higher speeds

The relationship between speed and total mass in these results may seem contradictory. In humans, higher weights are usually associated with slower maximum locomotion speed. This is due to the fact that these weights lead to the requirement of higher power output from their muscles to produce the same amount of movement. In a robot, the constraints from the torque output for each joint may be high enough that they still enable the same amount of movement it can produce for lower masses. Furthermore, there is not an incentive for the particular locomotion controller used in this setup to optimize for torque output over the execution of the desired motion. Higher velocities for robots with higher masses are associated with a higher torque output in every case of these experiments, as one can observe in Table 5.25. Higher forces applied to the center of mass as a result of the joints' torques may result in higher momentums than observed in lighter robots, leading to higher displacements.

5.3.4 Conclusions

These are the main conclusions reached from the experiments related to the XDE-manikin, that result from the discussion in Section 5.3.3.

- Taller models of the XDE-manikin reached higher mean speeds. These models can produce higher step lengths, also resulting in higher torque outputs.
- Heavier models also reached higher speeds, in general. This may be caused by higher momentums produced in its limbs, and also results in higher torque outputs.

5.4 Discussion

In this chapter we presented the results of experiments on optimizing the locomotion behavior of three different robots in different conditions: the iCub in floors with different frictions, the DARwIn-OP on ramps with different slopes, and the XDE-manikin with different heights and weights.

Impact of stability features

Multiple optimization setups were experimented, which had different sets of parameters optimized, and different locomotion features as optimization objectives. Two of the locomotion features were used as performance indicators for all these setups: the mean speed of the robot, and the total torque output in the locomotion. The other features served as stability measures, with the intent of reducing the risk of the robot falling. Adding these types of measures to the iCub optimization lead to a decrease in failure rate, and an increase of high speed behaviors, although these came at the cost of higher torque outputs. Higher stability leads to higher speeds, but a higher torque output is required to sustain both these gains.

Comparing the obtained performances with the literature

When comparing the obtained speeds with current literature, one must take into account the context of the comparisons. Is the trial done in simulation, or in the real robot? Is the terrain comparable? Is the control architecture the same, or similar? This last point is especially important for this work, since we propose an optimization framework, not the locomotion controller itself. This makes the comparison with speed values reached for that specific controller more important that the overall best value achieved. Speeds for the iCub and the DARwIn-OP showed improvement over references found for similar contexts, although only one could be found for iCub, and two for the DARwIn-OP.

Tuning the optimization setups

The initial experiments with the iCub, the DARwIn-OP, and the XDE-manikin show that some features of the locomotion of different robot models and controllers can be optimized with an EA. Adding and removing parameters from the iCub optimizations in further experiments shows how these can be further tuned. In the DARwIn-OP preliminary optimization, this fine tuning was further explored by using a large number of parameters for the optimization and analyzing their correlation with the optimization results to choose which ones to keep. The correlation analysis led to the possibility of an optimization with fewer parameters, a population with fewer solutions, and a lower number of generations. The speed result had a significant increase, regarding the optimization done on the floor with no ramp, while the torque indicator had a slightly higher (*i.e.* worse) value. The fact that this decision was made from a larger pool of parameters to begin with has the advantage of not loosing on potential beneficial sets of parameters.

Optimizing for different floor conditions

Repeating the iCub optimizations for different values of friction coefficient, and the DARwIn-OP optimizations for different floor slopes, showed that solutions for the optimizations perform differently when changing these conditions, which implies the need to know this information, and to take it into account, when choosing the best set of parameters for each occasion. Sensitivity analysis on the pareto-optimal fronts of these optimizations, in which the solutions from these fronts were tested in multiple conditions, showed that no single optimization was optimal for every situation.

Effects of the CoF on optimization results

In the second stage of the iCub experiments, higher values of CoF cause less restrictions on the forces applied by the feet on the ground, and allowed for higher speeds of locomotion. In the sensitivity analysis, sets of solutions optimized for lower CoFs were more successful in a broader range of values of CoF. The Pareto fronts optimized for higher values of CoF struggle with lower values of friction, that do not allow some of these behaviors without causing the robot to fall. These are, however, better at reaching higher speeds for those higher values of CoF, whereas solutions for floors with more constraints result in lower performance.

Effects of the floor slope on optimization results

Optimizations on the DARwIn-OP for floors with different ramps show that there is a range of values for the ramp's slope, around the middle of the range of values tested, where the fronts of solutions perform better, overall. The fronts optimized for values around 2 to 8 degrees, with 0 and 12 being the extremes of the tests, perform better across the multiple scenarios, in the percentage and number of successes with a minimum speed, maximum speed, and lowest torque output values.

A hypothesis for the mid-range optimization fronts performing better

The hypothesis for solutions optimized for a mid-range of values to show better average performances is that, since they have a shorter average numerical distance to the rest of the slope values, the relationship between the slope and the behavior outputs is somewhat linear. There are other factors that come into play, however, such as the steeper ramps being inherently riskier to traverse due to the effect of gravity, and lower values of the floor's friction being more likely to cause the robot to fall. In the case of the ramps in the DARwIn's case, this causes all fronts to have poor results for the highest slopes (10 and over). The fronts for lower values of slope, because they perform well in those lower slopes, have higher rates of success. When it comes to the floor's friction, lower values are riskier, which means the behaviors optimized for those will be more consistent across the board. These are not penalized for the high friction values, as opposed to the ramps, where they become a physical obstacle hard to traverse for solutions from different slopes. They are, however, penalized in terms of speed performance, where values for the middle range of frictions produce better average results.

Experiments with the XDE-manikin

Experiments on manikin models with the same kinematic structure, but scaled for different height/mass combinations, provide information on how these different sizes result in optimizations with different optimal solutions and performances. The highest speeds reached are much higher than the iCub's and DARwIn-OP's because of their relatively short height (1 m and 0.4545 m, respectively).

Importance of optimizing for different sizes

The effect of changes in height and mass of the robot in its locomotion could be important for its design phase, with the possibility of choosing these quantities according to what the desired performance resulting from the task is. Additionally, uncertainties in the physical model parameters of the robot, such as lengths and masses of its segments, are common, and it is therefore desirable to have a framework robust to these changes.

Chapter 6

An adaptive approach to humanoid locomotion

Building on the foundation of the optimization framework, we propose a method that uses the information from several simulations to adapt the locomotion task of a robot to environment changes. The data from the exploration phase, described in Chapter 4, is used to, first, identify the value of the environment variable, and then select the most appropriate solution, taking into account the user's requirements towards the locomotion features that result from that solution. This process should happen in the safest way possible, *i.e.*, conducting trials that do not cause the robot to fall, and as quickly as possible.

This chapter begins by stating the problem it is based on in Section 6.1, and then gives an overview of the framework used (Section 6.2). The way each function from the framework works is explained in Section 6.3, along with accompanying algorithms detailing them.

6.1 Problem definition

Considering a locomotion environment with an unknown value of the variable modeled in the exploration phase, $\boldsymbol{\theta}_{\text{new}}$, the objective is to find a solution $\boldsymbol{x}^*$, consisting of a vector of controller parameters' values, that optimizes the locomotion task towards a set of user requirements $\mathcal{P}$, consisting of a hierarchy of performance features, and a number associated with each of these features indicating how strongly the solution should be geared towards it.

6.2 Adaptation framework overview

The adaptation framework proposed tries to find an optimal solution supported by the information gathered from the exploration phase, described in Chapter 4. This is done in two stages: the identification of the value of the environment variable, and the choice

of the optimal solution, supported by the identification.

The identification phase has the objective of estimating the environment variable of the locomotion, $\boldsymbol{\theta}_{\text{new}}$. This variable was modeled in the exploration phase, from where $\mathcal{D}_{\text{s}}$ is obtained: a set of solutions $\mathcal{X}_{\text{s}}$ and their respective resulting behaviors $\mathcal{B}_{\text{s}}$ for different values of $\boldsymbol{\theta}$. This dataset contains the optimal solutions from optimizations on different environments, and their locomotion behavior, not only on the environment they were optimized for, but in others as well. This set of environments is referred to as Θ_{s}.

The identification is done by conducting a locomotion trial with a solution with known behaviors (also referred to as features) in multiple contexts, and then comparing the outcome of that trial with those behaviors. This is done in two steps; first, the solution $\boldsymbol{x}_{\text{id}}$, used for the identification trial, is selected. This set of parameters should both provide a safe behavior in as many different contexts as possible, and also allow for the correct identification of the context value. Conducting a trial with that solution, using the same conditions used in the optimizations for the exploration phase, results in the locomotion features, $\mathbf{b}_{\text{new}}$. These are used to identify the context value, which is referred to as $\boldsymbol{\theta}_{\text{id}}$. This identification is also supported by the information gathered from the sensitivity analysis from the exploration phase, contained in $\mathcal{D}_{\text{s}}$.

The last step of the adaptation is the choice of the optimal solution for the overall problem. This is done with the support of the information from the environment identification, $\boldsymbol{\theta}_{\text{id}}$, and the datasets of solutions and behaviors from the sensitivity analysis, $\mathcal{D}_{\text{s}}$. $\boldsymbol{\theta}_{\text{id}}$ is used to filter the list of solutions from $\mathcal{D}_{\text{s}}$ to the ones that were conducted in the new environment, and, from that new list, the solution that maximizes the requirements expressed in $\mathcal{P}$ is selected as the final solution to the problem, $\boldsymbol{x}^*$.

Figure 6.1 shows an overview of this framework. The details of each function are described in the next section.

6.3 Functions of the adaptation framework

This section outlines how each function of the adaptation system was implemented. Algorithm 4 details the procedure for the overall adaptation process, with further sections exposing and detailing each of its steps.

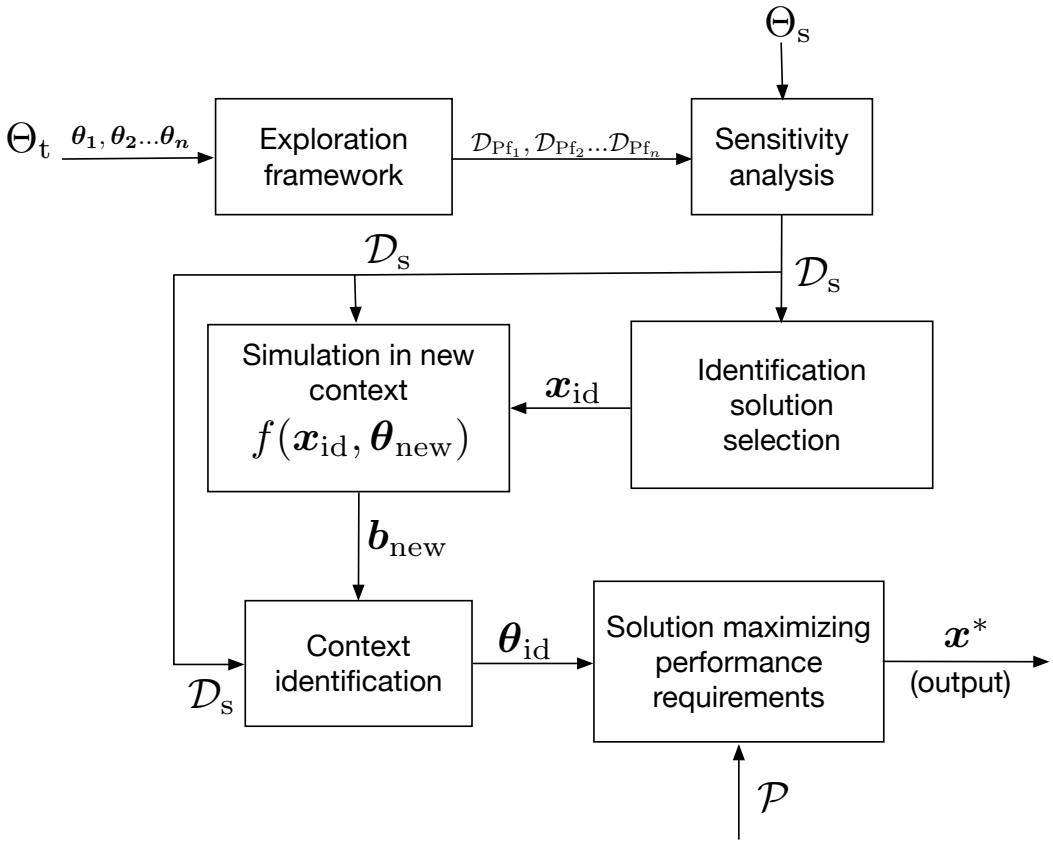


Figure 6.1: Architecture of the adaptation framework. The exploration framework (Figure 4.1) is used to optimize the locomotion of a robot model in $\boldsymbol{n}$ different environments, Θ_t, obtaining $\boldsymbol{n}$ different Pareto fronts datasets, $\mathcal{D}_{Pf_1}, \mathcal{D}_{Pf_2} ... \mathcal{D}_{Pf_n}$. The solutions from these datasets are then used in a sensitivity analysis (Figure 4.2) to test their performance in a set of different contexts, Θ_s. From the dataset resulting from that analysis, $\mathcal{D}_s$, a solution $\boldsymbol{x}_{id}$ is selected. This solution is tested in simulation in the unknown environment $\boldsymbol{\theta}_{new}$, in order to use the resulting locomotion features $\boldsymbol{b}_{new}$ to come up with $\boldsymbol{\theta}_{id}$, an estimation of $\boldsymbol{\theta}_{new}$. This estimation is used to filter the solutions list from $\mathcal{D}_s$, after which the solution, $\boldsymbol{x}^*$, that best fits the performance requirements from $\mathcal{P}$ will be selected.

Algorithm 4 Context adaptation procedure. Given a dataset $\mathcal{D}_\text{s}$, and a set of performance requirements $\mathcal{P}$, selects the solution $\boldsymbol{x}^*$ that better adapts to a new environment $\boldsymbol{\theta}_\text{new}$, while fitting the performance requirements. The cutoff values $\mathbf{c}_\text{s}$ and $\mathbf{c}_\text{d}$ can be changed to balance how much the identification solution selection prioritizes safety or features dispersion.

procedure CONTEXT_ADAPTATION($\mathcal{D}_\text{s}, \mathcal{P}, \boldsymbol{\theta}_\text{new}, \mathbf{c}_\text{s}, \mathbf{c}_\text{d}$)
  $\boldsymbol{x}_\text{id} \leftarrow \emptyset$ ▷ Solution to be used in the identification test.
  $\mathcal{B}_\text{id} \leftarrow \emptyset$ ▷ Features of $\boldsymbol{x}_\text{id}$ in the contexts from $\mathcal{D}_\text{s}$.
  $\boldsymbol{x}_\text{id}, \mathcal{B}_\text{id} \leftarrow$ SELECT_ID_SOLUTION($\mathcal{D}_\text{s}, \mathbf{c}_\text{s}, \mathbf{c}_\text{d}$)
  $\boldsymbol{b}_\text{new} \leftarrow \emptyset$ ▷ Features of $\boldsymbol{x}_\text{id}$ in the unknown environment.
  $\boldsymbol{b}_\text{new} \leftarrow f(\boldsymbol{x}_\text{id}, \boldsymbol{\theta}_\text{new})$ ▷ Run a identification test in the new environment.
  $\boldsymbol{b}_\text{new} \leftarrow$ NORMALIZE_FEATURES($\mathcal{D}_\text{s}, \boldsymbol{b}_\text{new}$) ▷ Normalize the values of $\boldsymbol{b}_\text{new}$ and $\mathcal{B}_\text{id}$,
  $\mathcal{B}_\text{id} \leftarrow$ NORMALIZE_FEATURES($\mathcal{D}_\text{s}, \mathcal{B}_\text{id}$) ▷ taking into account the features from $\mathcal{D}_\text{s}$.
  $\boldsymbol{\theta}_\text{id} \leftarrow 0$ ▷ Estimation of the value of the unknown context, $\boldsymbol{\theta}_\text{new}$.
  $\boldsymbol{\theta}_\text{id} \leftarrow$ IDENTIFY_CONTEXT($\boldsymbol{x}_\text{id}, \mathcal{B}_\text{id}, \boldsymbol{b}_\text{new}$)
  $\mathcal{D}_\text{id} \leftarrow$ GET_CONTEXT_SOLUTIONS($\mathcal{D}_\text{s}, \boldsymbol{\theta}_\text{id}$) ▷ Get the solutions related to the identified context value.
  $\boldsymbol{x}^* \leftarrow \emptyset$ ▷ Final solution to the context adaptation.
  $\boldsymbol{x}^* \leftarrow$ SELECT_SOLUTION($\mathcal{D}_\text{id}, \mathcal{P}$)
  $\boldsymbol{b}^* \leftarrow \emptyset$ ▷ Behavioral feature for $\boldsymbol{x}^*$ in the unknown context.
  $\boldsymbol{b}^* \leftarrow f(\boldsymbol{x}^*, \boldsymbol{\theta}_\text{new})$
end procedure

6.3.1 Selection of the solution for the identification process

The identification process requires a solution $\boldsymbol{x}_{\text{id}}$ that is both safe in the largest number of different environments possible, and also enables the identification of the environment variable in the new context, $\boldsymbol{\theta}_{\text{new}}$. To comply with this last point, a solution is selected so that it shows as much dispersion as possible in the resulting locomotion features, in the different contexts it is tested.

This dispersion will lead to an easier identification of $\boldsymbol{\theta}_{\text{id}}$. If there is more variation between these results, it is easier to differentiate between them, making it so the resulting behavior from the identification trial, $\boldsymbol{b}_{\text{id}}$, is easier to match to the correct context. If the solution had the same features in every context tested, for instance, it would be impossible to make an identification using this method.

The index of dispersion of a distribution is defined as,

$$\text{D} = \frac{\boldsymbol{\sigma}^2}{\boldsymbol{\mu}}, \tag{6.1}$$

with $\boldsymbol{\sigma}$ being the standard deviation, and $\boldsymbol{\mu}$ the mean of the distribution. Each solution from $\mathcal{X}_{\text{s}}$ is scored according to the dispersion the features of the locomotion (*e.g.* speed, torque) showed in the different contexts they were tested in. The dispersions from the different features are then added up into a single value. The features from locomotion that resulted in a fall were not considered.

The choice of the identification solution is made taking into account both the number of successful simulations observed in the different environments, and the total dispersion scored by the solution. To balance these two factors, two cutoff values are used, $\mathbf{c}_{\text{d}}$ and $\mathbf{c}_{\text{s}}$. $\mathbf{c}_{\text{d}}$ is the minimum required for the ratio between the dispersion of the chosen solution ($\mathbf{d}$) and the highest overall dispersion ($\mathbf{d}_{\text{max}}$) to be,

$$\frac{\mathbf{d}}{\mathbf{d}_{\text{max}}} \geq \mathbf{c}_{\text{d}}. \tag{6.2}$$

$\mathbf{c}_{\text{s}}$ represents the minimum number of successes the identification solution should have.

Finding a balance between $\mathbf{c}_{\text{d}}$ and $\mathbf{c}_{\text{s}}$ is important. Choosing the solution with the highest dispersion does not matter if it can only perform in two different contexts, while simply choosing a solution that is successful in every environment tested can make it useless in terms of differentiating between them. Our approach was to select the solution that respects both cutoffs and shows the highest dispersion. In case one does not exist, we choose one that has the highest number of successes, and, from those, the one that shows the highest dispersion. This approach is biased towards the number of simulations without the robot falling, and, while it served its purpose in these trials, others could be used. A more balanced one would be, for example, to successively lower both cutoffs (*e.g.*, $\mathbf{c}_{\text{s}}$ by 1 and $\mathbf{c}_{\text{d}}$ by 0.1) until a suitable solution is found.

Algorithms 5 and 6 show an overview of the functions that were used in this process.

Algorithm 5 Identification solution selection. From the sensitivity analysis dataset, $\mathcal{D}_\text{s}$, the algorithm tries to find a solution that respects both the cutoffs for the minimum ratio between the chosen solution's dispersion and the maximum observed, $\mathbf{c}_\text{d}$, and the number of successful solutions in different contexts, $\mathbf{c}_\text{s}$. The list of behaviors of that solution in different contexts is also returned.

function SELECT_ID_SOLUTION($\mathcal{D}_\text{s}$, $\mathbf{c}_\text{s}$, $\mathbf{c}_\text{d}$)
  $\boldsymbol{x}_\text{id} \leftarrow \emptyset$
  $\text{D}_{\mathcal{X}_\text{s}} \leftarrow \emptyset$ ▷ List with the total dispersion of each solution.
  $\text{D}_{\mathcal{X}_\text{s}} \leftarrow$ GET_FEATURES_DISPERSIONS($\mathcal{D}_\text{s}$)
  $\text{D}_\text{max} \leftarrow \emptyset$ ▷ Maximum dispersions for a given number of successes.
  $\mathcal{X}_\text{max} \leftarrow \emptyset$ ▷ Solutions associated with each dispersion from D_max.
  $\text{D}_\text{max}, \mathcal{X}_\text{max} \leftarrow$ ORDER_MAX_DISPERSIONS($\mathcal{D}_\text{s}, \text{D}_{\mathcal{X}_s}$)
  $\mathbf{d}_\text{max} \leftarrow$ MAXIMUM(D_max) ▷ Overall maximum dispersion.
  for i $\leftarrow$ LENGTH(D_max) to 1 **do** ▷ Start at the highest number of successes.
    $\mathbf{d} \leftarrow \text{D}_{\text{max},i}$ ▷ Best dispersion for **i** number of successes.
    $\boldsymbol{x} \leftarrow \mathcal{X}_{\text{max},i}$ ▷ Solution showing the best dispersion.
    if i $< \mathbf{c}_\text{s}$ **then** ▷ If there are no solutions with the cutoff
      $\boldsymbol{x}_\text{id} \leftarrow \boldsymbol{x}$ ▷ number of successes, return best solution for the
      break ▷ maximum number of successes observed.
    end if
    if $\frac{\mathbf{d}}{\mathbf{d}_\text{max}} \geq \mathbf{c}_\text{d}$ **then** ▷ If the solution's dispersion respects
      $\boldsymbol{x}_\text{id} \leftarrow \boldsymbol{x}$ ▷ the cutoff, select it as the ID solution.
      break
    end if
  end for
  $\mathcal{B}_\text{id} \leftarrow \emptyset$ ▷ List of features for the chosen $\boldsymbol{x}_\text{id}$ solutions for each context in Θ_s.
  $\mathcal{B}_\text{id} \leftarrow$ GET_SOLUTION_FEATURES($\boldsymbol{x}_\text{id}$)
  return $\boldsymbol{x}_\text{id}, \mathcal{B}_\text{id}$
end function

Algorithm 6 Sub-functions used in the identification solution selection algorithm. GET_FEATURES_DISPERSIONS, given a dataset of tests $\mathcal{D}_\text{s}$, returns a list with the total dispersion in locomotion features shown by each solution, $\text{D}_{\mathcal{X}_\text{s}}$. ORDER_MAX_DISPERSIONS, given the same dataset $\mathcal{D}_\text{s}$, and the list of dispersions $\text{D}_{\mathcal{X}_\text{s}}$, returns a list with the best dispersion for a given number of successes, and another with the solutions that result in those dispersions.

function GET_FEATURES_DISPERSIONS($\mathcal{D}_\text{s}$)
  $\text{D}_{\mathcal{X}_\text{s}} \leftarrow \emptyset$ ▷ List of dispersions of the features from $\mathcal{D}_\text{s}$.
  for $i \leftarrow 1$ to LENGTH($\mathcal{D}_\text{s}$) **do** ▷ Cycle through behavior results from different solutions.
    $\mathcal{B} \leftarrow \mathcal{B}_{\text{s},i}$ ▷ List of behavior for different contexts, for the current solution.$\mathcal{B}_\text{s} \in \mathcal{D}_\text{s}$.
    $\mathbf{d}_\text{total} \leftarrow 0$ ▷ Value of total dispersion for the current $\boldsymbol{b}_s$.
    for $j \leftarrow 1$ to number of columns in $\mathcal{B}$ **do** ▷ Each column is associated with one feature.
      $\mathcal{B}_j \leftarrow$ GET_COLUMN($\mathcal{B}, j$) ▷ Get all the values for feature j in the dataset.
      $\boldsymbol{\mu} \leftarrow$ AVERAGE($\mathcal{B}_j$)
      $\boldsymbol{\sigma} \leftarrow$ STANDARD_DEVIATION($\mathcal{B}_j$)
      $\mathbf{d}_j \leftarrow \frac{\boldsymbol{\sigma}^2}{\boldsymbol{\mu}}$ ▷ Dispersion for a single features
      $\mathbf{d}_\text{total} \leftarrow \mathbf{d}_\text{total} + \mathbf{d}_j$ ▷ Add $\mathbf{d}_j$ to the total.
    end for
    $\text{D}_{\mathcal{X}_\text{s}}$.insert($\mathbf{d}_\text{total}$) ▷ Insert the total dispersion of one solution in the list.
  end for
  return $\text{D}_{\mathcal{X}_\text{s}}$
end function

function ORDER_MAX_DISPERSIONS($\mathcal{D}_\text{s}$, $\text{D}_{\mathcal{X}_\text{s}}$)
  $\text{D}_\text{max} \leftarrow \emptyset$ ▷ List to hold the maximum dispersions for number of successes.
  $\mathcal{X}_\text{max} \leftarrow \emptyset$ ▷ List to hold the solutions that show the maximum dispersions.
  $\text{S} \leftarrow$ GET_NR_SUCCESSES($\mathcal{D}_\text{s}$) ▷ List with number of successes for each solution.
  for $i \leftarrow 1$ to LENGTH($\mathcal{D}_s$) **do** ▷ Loop through each solution, selecting
    $\mathbf{d} \leftarrow \text{D}_{\mathcal{X}_\text{s},i}$ ▷ its dispersion,
    $\mathbf{j} \leftarrow \text{S}_i$ ▷ the number of successes in different contexts
    $\mathbf{x} \leftarrow \mathcal{X}_{\text{s},i}$ ▷ and the solution itself. $\mathcal{X}_\text{s} \in \mathcal{D}_\text{s}$
    if $\text{d} > \text{D}_{\text{max,s}}$ **then** ▷ If current dispersion is greater than the maximum,
      $\text{D}_{\text{max},j} \leftarrow \boldsymbol{d}$ ▷ for the number of successes the solution has across contexts.
      $\mathcal{X}_{\text{max},j} \leftarrow \boldsymbol{x}$ ▷ Assign the solution and dispertion as the best
    end if ▷ for the current number of successes.
  end for
  return $\text{D}_\text{max}, \mathcal{X}_\text{max}$
end function

6.3.2 Identification of the new context

The goal of the identification phase is to choose an adequate value for an unknown locomotion context that was modeled as a numerical variable (*e.g.*, a friction value). This value, $\boldsymbol{\theta}_{\text{id}}$, works as an approximation of $\boldsymbol{\theta}_{\text{new}}$, and will be used to inform the decision of selecting the final solution for the problem.

After determining $\boldsymbol{x}_{\text{id}}$, the solution to be used in the identification, a test of the locomotion behavior is conducted in the unknown environment, with the same setup (*i.e.*, robot model, control framework) as the exploration phase,

$$\boldsymbol{b}_{\text{new}} = f(\boldsymbol{x}_{\text{id}}, \theta_{\text{new}}). \tag{6.3}$$

The features that resulted from the test, $\boldsymbol{b}_{\text{new}}$, can then be compared with the features that $\boldsymbol{x}_{\text{id}}$ shows in simulations in the different environments Θ_{s}, from the dataset $\mathcal{D}_{\text{s}}$, which are grouped in the set $\mathcal{B}_{\text{id}}$.

The distance between $\boldsymbol{b}_{\text{new}}$ and each vector of features from $\mathcal{B}_{\text{id}}$ can show which of the sensitivity tests the identification test is closest to, and, therefore, which context value the new environment is more likely to approximate to. This comparison can be done with different distance metrics, the most common being the Euclidean distance. Considering two points in a Euclidean space [1] with $\boldsymbol{n}_b$ dimensions (corresponding to the numbers of features), $\boldsymbol{b}_{\text{new}} = (\boldsymbol{b}_{\text{new},1}, \boldsymbol{b}_{\text{new},2}, \ldots, \boldsymbol{b}_{\text{new},n_b})$ and $\boldsymbol{b}_{\text{id}} = (\boldsymbol{b}_{\text{id},1}, \boldsymbol{b}_{\text{id},2}, \ldots, \boldsymbol{b}_{\text{id},n_b})$, this distance is calculated as

$$\textbf{distance}_{\text{Euclidean}}(\boldsymbol{b}_{\text{new}}, \boldsymbol{b}_{\text{id}}) = \sqrt{(\boldsymbol{b}_{\text{id},1} - \boldsymbol{b}_{\text{new},1})^2 + \ldots + (\boldsymbol{b}_{\text{id},n_b} - \boldsymbol{b}_{\text{new},n_b})^2}. \tag{6.4}$$

The Euclidean distance measures the distance in a straight line between two points, giving equal weight to all dimensions in their contribution to the value. This means, in this case, that every feature will have a similar impact, given that they have a similar difference in value between the two vectors. Since there is no indication of whether some of these features better differentiate between contexts, it is desirable that they have similar impacts, regardless of their numeric scale. In order to achieve this, every feature value is normalized between 0 and 1. Given a single feature value $\boldsymbol{b}_i$, the maximum value of that feature on the entire sensitivity dataset $\mathcal{B}_{\text{s}}$, $\max_{b_i}$, and its minimum, $\min_{b_i}$, the normalization is conducted as follows

$$\boldsymbol{b}_{i,\text{norm}} = \frac{\boldsymbol{b}_i - \min_{b_i}}{\max_{b_i} - \min_{b_i}}. \tag{6.5}$$

After calculating all the distances between $\boldsymbol{b}_{\text{new}}$ and the vectors from $\mathcal{B}_{\text{id}}$, the context that corresponds to the vector that shows the shortest distance, i.e., the closest features, is chosen as an estimate, and used as $\boldsymbol{\theta}_{\text{id}}$.

[1] The Euclidean n-space is also called Cartesian space, and it is the space of all n-tuples of real numbers.

Algorithm 7 shows the process used for this identification step. The normalization of the features is included in the overall adaptation Algorithm 4, as the NORMALIZE_FEATURES function.

Algorithm 7 Context identification algorithm. Given a solution $\boldsymbol{x}_{\text{id}}$, compares its features from the sensitivity tests, $\mathcal{B}_{\text{id}}$, with the ones from the test in the new environment, $\boldsymbol{b}_{\text{new}}$, and returns the context value, θ_{id}, which simulation shows the closest features' values.

```
function IDENTIFY_CONTEXT(x_id, B_id, b_new)
    distances ← ∅                                   ▷ Distances between features sets.
    for i ← 1 to LENGTH(B_id) do                    ▷ Loop through the x_id features for each context.
        b ← B_id,i                                  ▷ Features for the current context.
        distance ← EUCLIDEAN_DISTANCE(b_new, b)
        distances.insert(distance)
    end for
    index ← ARG_MIN(distances)                      ▷ Index of the minimum distance between features.
    θ_id ← θ_index                                  ▷ Context value for the shortest distance.
    return θ_id
end function
```

6.3.3 Selecting the final solution

The solution to the main problem, $\boldsymbol{x}^*$, is one that needs to both lead the robot to walk in the new environment without falling, and follows the performance requirements from $\mathcal{P}$. The solution is selected from a list which is taken from $\mathcal{D}_{\text{s}}$ by removing the solutions and features pairs that are not relative to tests in $\boldsymbol{\theta}_{\text{id}}$. This dataset is called $\mathcal{D}_{\text{id}}$.

The performance requirements set contain a vector related to the hierarchy of the desired features, and another related to the weight given to each of these features, $\mathcal{P} = \mathbf{h}_{\text{p}}, \mathbf{w}_{\text{p}}$. The hierarchy vector contains the order of preference given to each feature. The weights vector contains a number from 0 to 1.0, with higher values translating into more solutions that do not perform well in that given feature being discarded. Both vectors' size is $\boldsymbol{n}_b$, the number of features of locomotion considered in the setup.

The selection process works in $\boldsymbol{n}_b + 1$ steps. At each step, the list of solutions, and respective behaviors, $\mathcal{D}_{\text{id}}$, is filtered by removing solutions that do not perform well in regard a given feature. For the step i, the feature indicated by the hierarchy $\mathbf{h}_{\text{p},i}$ is considered. A given solution is removed from the list if its value for that feature, $\boldsymbol{b}_i$, is below a certain fraction of the best value in the list, $\max_{b,i}$,

$$\boldsymbol{b}_i < \max_{b,i} \times \mathbf{w}_{\text{p},i}. \tag{6.6}$$

If the feature in question is to be minimized, instead of maximized,

$$\boldsymbol{b}_i > \min_{b,i} \times \frac{1}{\mathbf{w}_{\text{p},i}} \tag{6.7}$$

Table 6.1: Set of solutions and behaviors for an example $\mathcal{D}_{\text{id}}$ dataset. Each line contains one solution $\boldsymbol{x}$ from $\mathcal{X}_{\text{id}}$, and its respective behavior vector $\boldsymbol{b}$ from $\mathcal{B}_{\text{id}}$. The context $\boldsymbol{\theta}$ of the solution test is $\boldsymbol{\theta}_{\text{id}}$ for every line. The first two features were maximized, where the third was minimized.

$\mathcal{X}_{\text{id}}$	$\mathcal{B}_{\text{id}}$		
$\boldsymbol{x}_1$	0.50	1.20	0.02
$\boldsymbol{x}_2$	0.43	2.11	0.11
$\boldsymbol{x}_3$	0.70	3.24	0.06
$\boldsymbol{x}_4$	0.12	2.50	0.07
$\boldsymbol{x}_5$	0.91	0.95	0.06
$\boldsymbol{x}_6$	1.20	2.70	0.09
$\boldsymbol{x}_7$	0.20	2.33	0.10
$\boldsymbol{x}_8$	0.11	1.89	0.03
$\boldsymbol{x}_9$	0.40	1.05	0.05
$\boldsymbol{x}_{10}$	1.14	0.80	0.06

is the inequation used to decide whether that solution is removed.

After the $\boldsymbol{n}_b$ stages of filtering the solutions list are done, the final solution is selected. This selection is done by taking the first feature of the hierarchy vector, $\mathbf{h}_{\text{p},1}$ (*i.e.*, the feature given the highest priority), and choosing the solution that optimizes the performance towards that behavior.

Example of the selection towards $\mathcal{P}$

Here is displayed a small example of the solution selection procedure from Section 6.3.3. Table 6.1 shows an example dataset with 10 solutions, and 3 features measured in the locomotion process. Table 6.2 shows the solution's list at each of the four stages of the solution selection process, when using the performance requirements $\mathbf{h}_{\text{p}} = (b_2, b_1, b_3)$ and $\mathbf{w}_{\text{p}} = (0.5, 0.3, 0.6)$.

In the first stage, the second feature is considered ($\mathbf{h}_{\text{p},1} = b_2$). Since this feature was maximized during the exploration phase, the first stage of Table 6.2 shows the solutions ordered from the highest to the lowest value of that feature. Since the weight for that feature is $\mathbf{w}_{\text{p},1} = 0.5$, the solutions removed are the ones in which $\boldsymbol{b}_1 \leq \max_{b,1} \times \mathbf{w}_{\text{p},1} = 3.24 \times 0.5 = 1.62$. This leaves the bottom four solutions out of future stages.

For the second stage, six solutions remain, and the focus is on the first feature, which was maximized. The minimum value of the feature is $1.20 \times 0.3 = 0.36$, which brings the list down to three solutions. In the third stage the removal is due to the performance in the third feature, which was minimized. The cutoff value is of $0.06 \times \frac{1}{0.6} = 0.1$, meaning one of the solutions is removed.

The first three stages end up with a list of two solutions. The choice between them is done according to the feature given first priority, $\mathbf{h}_{\text{p},1} = b_2$. The solution with the best (in this case, highest) value for that given features, $\boldsymbol{x}_3$, is selected, with the rest being removed.

Table 6.2: Process of solution selection using the $\mathcal{D}_{\text{id}}$ from Table 6.1, $\mathbf{h}_{\text{p}} = (b_2, b_1, b_3)$, and $\mathbf{w}_{\text{p}} = (0.5, 0.3, 0.6)$. Four stages of solutions removal are shown, each separated by an horizontal line. The rows corresponding to the solutions removed at each stage are colored in gray.

$\mathcal{X}_{\text{id}}$	$\mathcal{B}_{\text{id}}$		
$\boldsymbol{x}_3$	0.70	3.24	0.06
$\boldsymbol{x}_6$	1.20	2.70	0.09
$\boldsymbol{x}_4$	0.12	2.50	0.07
$\boldsymbol{x}_7$	0.20	2.33	0.10
$\boldsymbol{x}_2$	0.43	2.11	0.11
$\boldsymbol{x}_8$	0.11	1.89	0.03
$\boldsymbol{x}_1$	0.50	1.20	0.02
$\boldsymbol{x}_9$	0.40	1.05	0.05
$\boldsymbol{x}_5$	0.91	0.95	0.06
$\boldsymbol{x}_{10}$	1.14	0.80	0.06
$\boldsymbol{x}_6$	1.20	2.70	0.09
$\boldsymbol{x}_3$	0.70	3.24	0.06
$\boldsymbol{x}_2$	0.43	2.11	0.11
$\boldsymbol{x}_7$	0.20	2.33	0.10
$\boldsymbol{x}_4$	0.12	2.50	0.07
$\boldsymbol{x}_8$	0.11	1.89	0.03
$\boldsymbol{x}_3$	0.70	3.24	0.06
$\boldsymbol{x}_6$	1.20	2.70	0.09
$\boldsymbol{x}_2$	0.43	2.11	0.11
$\boldsymbol{x}_3$	0.70	3.24	0.06
$\boldsymbol{x}_6$	1.20	2.70	0.09

Algorithm 8 Solution selection algorithm. Given the dataset for the sensitivity analysis in the exploration phase, and considering only the tests relative to θ_{id}, $\mathcal{D}_{\text{id}}$, this function returns the solution $\boldsymbol{x}^*$ that better fits the performance requirements $\mathcal{P}$.

function SELECT_SOLUTION($\mathcal{D}_{\text{id}}, \mathcal{P}$)
  $\boldsymbol{x}^* \leftarrow \emptyset$ ▷ Final solution of the framework.
  $\mathbf{h}_{\text{p}}, \mathbf{w}_{\text{p}} \leftarrow \mathcal{P}$ ▷ Features' hierarchy and weights.
  $\mathcal{X}, \mathcal{B} \leftarrow \mathcal{D}_{\text{id}}$ ▷ Identification set solutions and respective behaviors.
  nr_solutions = LENGTH($\mathcal{D}_{\text{id}}$)
  nr_features = LENGTH($\boldsymbol{b}$) ▷ For any given $\boldsymbol{b} \in \mathcal{B}_{\text{s}}$.
  $\text{Opt}_b \leftarrow \emptyset$ ▷ Optimum value for each feature in the entire dataset.
  for $i \leftarrow 1$ to nr_features **do**
    $\mathcal{B}_i \leftarrow$ GET_COLUMN($\mathcal{B}, i$) ▷ Get all the values for feature i in the dataset.
    if feature i is maximized **then** ▷ If the feature was maximized,
      $\text{Opt}_{b,i} \leftarrow$ MAXIMUM($\mathcal{B}_i$) ▷ Get the maximum value of the entire set.
    else
      $\text{Opt}_{b,i} \leftarrow$ MINIMUM($\mathcal{B}_i$) ▷ Otherwise get the minimum.
    end if
    for $j \leftarrow 1$ to nr_solutions **do**
      $\boldsymbol{x} \leftarrow \mathcal{X}_j$ ▷ Current solution evaluated.
      $\boldsymbol{b} \leftarrow$ GET_ROW($\mathcal{B}, j$) ▷ Features vector for $\boldsymbol{x}$.
      if $\boldsymbol{b}_i$ is maximized **then** ▷ Check if the feature was maximized.
        cutoff $\leftarrow \mathbf{w}_{\text{p, i}} \times \text{Opt}_{b,i}$ ▷ Calculate the cutoff of the feature value.
        if $\boldsymbol{b}_i <$ cutoff **then** ▷ If the solution's performance is not good enough.
          $\mathcal{X}$.remove($\boldsymbol{x}$) ▷ Remove the solution and respective
          $\mathcal{B}$.remove($\boldsymbol{b}$) ▷ behavior from their lists.
        end if
      else ▷ Make the appropriate changes if the feature was minimized.
        cutoff $\leftarrow \frac{1}{\mathbf{w}_{\text{p, i}}} \times \text{Opt}_{b,i}$ ▷ Change the cutoff calculation.
        if $\boldsymbol{b}_i >$ cutoff **then** ▷ Inequality is reversed.
          $\mathcal{X}$.remove($\boldsymbol{x}$)
          $\mathcal{B}$.remove($\boldsymbol{b}$)
        end if
      end if
    end for
  end for
  $i \leftarrow \mathbf{h}_{\text{p, 1}}$ ▷ Choose the highest prioritized feature for the final selection.
  $\mathcal{B}_i \leftarrow$ GET_COLUMN($\mathcal{B}, i$) ▷ Get all the values for feature i in the dataset.
  if feature i is maximized **then**
    $j \leftarrow$ ARG_MAX($\mathcal{B}_i$) ▷ Return the index corresponding to the maximum value.
  else
    $j \leftarrow$ ARG_MIN($\mathcal{B}_i$) ▷ Return the index corresponding to the minimum value.
  end if
  $\boldsymbol{x}^* \leftarrow \mathcal{X}_j$ ▷ Solution, from the ones left, that optimizes the highest priority feature.
  return $\boldsymbol{x}^*$
end function

Chapter 7

Humanoid locomotion adaptation to unknown terrain features

This chapter contains the experiments conducted to test the adaptation framework from Chapter 6.

The locomotion of a model of the iCub robot was adapted to floors with different coefficients of friction, while a model of the DARwIn-OP was adapted to walk up ramps with different slope values.

7.1 Wilcoxon test

This section details a procedure used to compare results obtained during this chapter. It compares the performance values achieved from different methods, by evaluating the differences between them in each point tested.

The Wilcoxon signed-ranked test (Wilcoxon, 1945) compares two related samples of values by testing the hypothesis that the mean value of their populations do not differ significantly. It is applied to two populations with paired data, and calculates the difference between each of these pairs in order to analyze how different the populations are, overall.

7.1.1 Procedure

With $\boldsymbol{x}_{1,i}, \boldsymbol{x}_{2,i}$, being the measurements of the pairs $i = 1, \ldots, N$, the test considers only N_r pairs, excluding the ones where $|\boldsymbol{x}_{2,i} - \boldsymbol{x}_{1,i}| = 0$, which means $N_\mathrm{r} \leq N$. The test evaluates the null hypothesis H_0 that the difference between the pairs follows a symmetric distribution around 0, against the H_1 hypothesis that the differences do not follow that type of distribution.

The test procedure is as follows:

1. For $i = 1, \ldots, N$, calculate the module of the difference between the paired values, $|\boldsymbol{x}_{2,i} - \boldsymbol{x}_{1,i}|$, and $\mathrm{sign}(\boldsymbol{x}_{2,i} - \boldsymbol{x}_{1,i})$. The sign function returns 1 if the difference is positive, 0 if it is null, and -1 if it is negative.

2. Order the N_r pairs from smallest to largest absolute difference.

3. Assign a rank R_i to each difference, with the smallest one getting rank 1.

4. Calculate
$$W = \sum_{i=1}^{N_\mathrm{r}} [\mathrm{sign}(\boldsymbol{x}_{2,i} - \boldsymbol{x}_{1,i}) \times R_i], \tag{7.1}$$
which is the sum of the signed ranks, *i.e.*, the sum of the ranks of every difference, weighted by its sign.

5. Under H_0, W follows a distribution with an expected value of 0 and variance
$$\frac{N_\mathrm{r}(N_\mathrm{r}+1)(2 \times N_\mathrm{r}+1)}{6}. \tag{7.2}$$
A critical value from a reference table, $W_{\mathrm{critical},N_\mathrm{r}}$, can be used to evaluate whether W is within the distribution limits, and the null hypothesis H_0 is rejected if $|W| > W_{\mathrm{critical},N_\mathrm{r}}$.

A p-value can also be calculated using the sums of ranks R from the distributions of each population. This process is described in Pratt (1959). A small p-value rejects the null hypothesis that the differences between the pairs are due to chance, while bigger values contribute towards confirming H_0.

7.2 Adapting the iCub's locomotion control to different coefficients of friction

This experiment applied the framework presented in Chapter 6, and shown in Figure 6.1. The dataset of information used in this approach ($\mathcal{D}_\mathrm{s}$) is the one that resulted from the exploration phase and sensitivity analysis presented in Section 5.1.4.

The information from this dataset is used to estimate the value of the coefficient of friction of different floors, and that estimation is then used to select a solution that enables a locomotion behavior that suits predetermined performance requirements.

The adaptation method was compared to two others. In the naive approach, the solution that best fits the performance requirements is done without taking into account the floor's coefficient of friction at all. In the other one, the random approach, an estimation for the CoF is done randomly, and that value is then used to select the solution that best fits the requirements.

7.2.1 Adaptation framework setup

The dataset $\mathcal{D}_\mathrm{s}$ contains solutions to the optimization of a forward walk of a model of the iCub robot in different terrains. These solutions consist of values for seven different parameters of a locomotion controller which is based on the locomotion's dynamics and a task hierarchy (see Appendix B). When applied to a forward walk of the iCub robot,

a controller with these parameters results in four different features of locomotion that were chosen to be measured. The context variable modeled, in this case the CoF, affects the behaviors these solutions result in.

The seven parameters used in the optimization are shown in Table 5.4, and the four features measured in Table 5.5. Two of these features were performance related, the mean speed and torque output (τ_{total}), while the other two were added to increase the stability of the task (global average f.r. indicator and average trajectory error). The solutions from different optimizations were tested for the context values of 0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 1.25, and 1.5, from which resulted the sensitivity analysis dataset.

In these experiments, the adaptation framework was applied to $\mathcal{D}_{\text{s}}$ in order to adjust the locomotion to floors with CoF values ranging from 0.05 to 2.0, with 0.05 intervals (extremes included). The cutoff values used in the adaptation algorithm (see Section 6.3.1) were $\mathbf{c}_{\text{s}} = 5$, for the minimum number of successes across different floors, and $\mathbf{c}_{\text{d}} = 0.8$, for the minimum ratio of features dispersion relative to the maximum. Two different performance requirements were tested for: one solely focused on speed, and a balanced approach that more closely prioritizes torque output and speed. The first one had the hierarchy $\mathbf{h}_{\text{p}} =$ (mean speed), and weights $\mathbf{w}_{\text{p}} = (1.0)$, meaning it only takes speed into account when choosing the final solution, making the rest of the indicators' hierarchy and weights irrelevant. The balanced approach had $\mathbf{h}_{\text{p}} =$ (τ_{total}, mean speed, f.r. indicator, trajectory error), and $\mathbf{w}_{\text{p}} = (0.7, 0.7, 0.5, 0.5)$.

Two other approaches were used alongside the proposed framework, in order to verify how impactful are key aspects of the algorithm. The naive approach skips the identification of the environment variable and goes directly to the step of choosing the final solution, doing so without being informed on the context variable $\boldsymbol{\theta}$. This serves as a control for the possibility that $\boldsymbol{\theta}$ doesn't affect the locomotion behavior enough to justify the process of identification. The other approach is a random one, where the identification is done by selecting a random number for the floor friction. In this case, the number was selected from a list of the friction values used in the sensitivity analysis (2 paragraphs above), since these are the ones the adaptation algorithm also chooses from. This approach serves as control for the possibility that the identification does not provide a clear advantage over simply choosing a random value for the context variable.

7.2.2 Results

The results for these experiments uses a paired differences test, called the Wilcoxon test (Wilcoxon, 1945), described in Section 7.1. It compares the performance of each approach in a given indicator, for each unknown context value tested. A sum of ranks indicates how well each approach performed when compared with the other. A p-value ≤ 0.05 indicates there is strong evidence the approach with the best sum of ranks is better performing than the other one. A value greater than that is not considered statistically significant. A higher sum of ranks indicates better performance for the speed feature, while a lower one is desired for the torque output feature (since one seeks to minimize it).

Table 7.1: Wilcoxon test results for our approach (labeled ID) against random and naive approaches, for the speed and torque features. Results related to the adaptation tests on the iCub. The values associated with a statistical significant test (p-value $<$ 0.05) are bolded.

Performance requirements	Feature compared	Sum of the ranks of the differences		p-value
		ID approach	Random approach	
Balanced	Speed	**589**	221	**0.0143**
Balanced	Torque	558	252	0.0573
Speed	Speed	**595**	224	**0.0123**
Speed	Torque	430	389	0.8125
		ID approach	Naive approach	
Balanced	Speed	**820**	0	**0.0000**
Balanced	Torque	667	**153**	**0.0006**
Speed	Speed	**571**	243	**0.0261**
Speed	Torque	**146**	668	**0.0006**

Wilcoxon test speed results

Table 7.1 shows the results of a Wilcoxon test used to compare the adaptation framework to both the random and naive approaches. These results show the identification approach outperforms the other two in the speed feature, both for the balanced and speed focused performance requirements. The differences are all statistically significant, with the naive approach being outperformed for every context value, in the balanced performance requirements (it has a sum of ranks equal to 0). This indicates that the naive approach, ignoring the context value, always goes for a solution that does not perform well for the speed indicator for the current environment.

Wilcoxon test torque results

Regarding the torque output indicator, the identification approach performs worse in three out of the four cases, with only two of them being statistically significant: one in which it performs better, and other where the performance is worse. Both cases are observed in the naive approach comparison. This can happen because the solution the approach always chooses with the speed focused requirements, constant due to always ignoring $\boldsymbol{\theta}$, is one that has especially bad torque performance, while the one chosen in the balanced requirements has better performance regarding the torque output. Additionally, the balanced requirements that result in the naive approach having a better torque performance are the same ones that result in it having a sum of ranks of 0 for the speed indicator. This situation is bound to occur with performance requirements where the weights are closer to a balance between all the indicators, since the solutions for a given context can have varying spreads of performance, and varying

Table 7.2: Success rates and average values of performance indicators for the adaptation phase tests for the iCub. The best averages for each performance requirement are bolded.

Approach	Performance requirements	Success rate (%)	Average mean speed (m/s)	Average $\boldsymbol{\tau}_{\text{total}}$($N^2$.$m^2$.s)
ID	Balanced	**95.0**	**0.1020**	**422.44**
Random	Balanced	92.5	0.0936	567.28
Naive	Balanced	85.0	0.0672	453.46
ID	Speed	**80.0**	**0.2676**	**691.36**
Random	Speed	72.5	0.2563	716.65
Naive	Speed	55.0	0.2531	1102.63

best values, for each indicator. In the case of a set of requirements with full focus on one indicator, such as the speed focus on these experiments, this particular situation cannot occur.

Average performance values

Table 7.2 shows the performance averages of every approach, for both performance requirements tested. The adaptation framework shows better averages for the success rate (percentage of trials where the robot did not fall), the average mean speed, and the average torque output. Although the naive approach showed a better sum of ranks in the Wilcoxon test, for the torque indicator, it shows a higher (therefore worse) average. The Wilcoxon test ranks the difference between a given pair of values (each belonging to the performance of one approach in a given context), and gives a higher weight to the biggest differences. This weight is fixed though, which means that it may not accurately represent the absolute difference between performances for certain cases. In this situation, the naive approach has a lower (better) sum of ranks because it outperforms the identification approach more often than not, but it has a higher average torque output because, when it does perform worse, the differences are higher. Specifically, it performs worse in the tests the robot falls, which are more often in the naive approach, given its lower success rate.

The balanced set of requirements results in safer solutions

The identification process for the experiments for different performance requirements yields the same results, since these requirements do not affect it. This means that the difference in success rates for the adaptation framework for the different requirements is a result of the choice of the final solution. For the context values where the identification was not correct, the balanced requirements have a higher chance of choosing solutions that do not fail in these environments, while the speed focused ones choose solutions that are riskier and therefore more prone to fail in contexts with an incorrect value

estimation.

Differences between the random and naive approaches

The random approach has better success rates than the naive approach for both performance requirements. The naive approach, selecting the best performing solution with no regard for the context value, can end up choosing one that performs well in a certain context, but that has an overall low success rate across different environments in the sensitivity analysis. This can be evidenced in this particular instance with the values from Tables 5.7 to 5.10. Regarding the speed focused performance requirements, in Table 5.9 it is evident the best solution would be one optimized for the CoF of 0.75, since it achieved the maximum speed observed (0.2736 m/s). This solution, however, may not have a good success rate in other contexts, as shown in Table 5.7, where the Pareto front for 0.75 CoF has one of the lowest averages. A similar case can be made for the balanced performance requirements. The other approach, even selecting a context value at random, has a high probability of selecting a solution that performs well in a great variety of contexts, simply because the naive approach ends up selecting one of the worst.

Identification of context values θ

Figure 7.1 shows the results of the identification itself, for each context value tested. There is a discrepancy between the behaviors the algorithm can identify, given how it works, and the ones it is tested for. The values tested were between 0.05 and 2.00, with steps of 0.05, while the ones that can be identified are 0.05, 0.10, 0.25, 0.50, 0.75, 1.00, 1.25, and 1.50 (the values tested in the sensitivity analysis).

Following the hypothesis that the same solution produces similar behaviors for values of friction near each other, the desired behavior for the identification process is the one observed for the real context values ranging from 0.1 to around 1.25. In this section the identified values for each test are around the closest value the algorithm could estimate, *i.e.* the values which behaviors were known for each solutions. This means that, as an example, even though there was no information for the behavior the identification solution should produce for 0.55, the estimated value is still 0.50. This happens with most values in the highlighted section, with some of them (0.5 to 1.25) being the center for their closest neighbors, both to the left and right, as evidenced by the vertical lines.

There are some outliers that go against the hypothesis of an approximately linear relationship. The value estimated for 0.05 was 1.50, the farthest value that could be estimated. The vertical lines for 0.10 and 0.25 show that estimated values around them are not centered around these values, although they were identified as values that are directly above or below those. The values identified around 1.50 oscillate between 1.25 and 1.50, even those that are above it. Since the sensitivity analysis did not include values above 1.50, a linear relationship would mean that those were all estimated as 1.50.

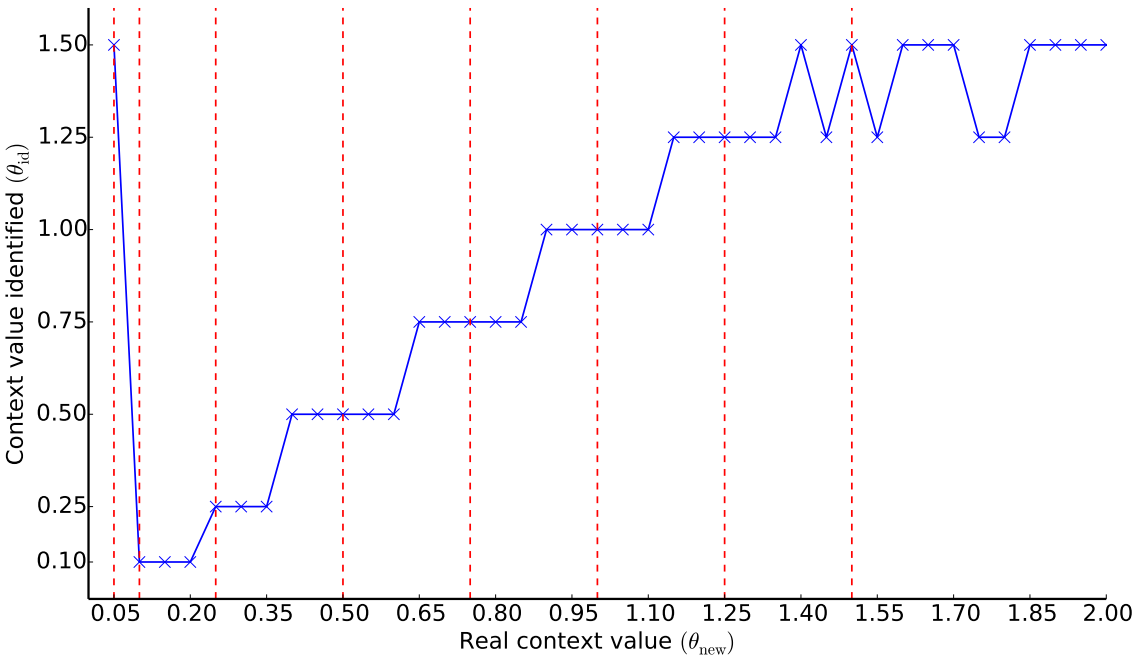


Figure 7.1: Real value of the context tested for, against the value identified by the adaptation framework. Results for the experiments with the iCub on floors with different CoF. Vertical lines mark the context values which behaviors the solutions were tested for in the sensitivity analysis.

7.2.3 Conclusions

Here the main conclusions from the adaptation experiments related to the iCub, which were reached in Section 7.2.2 are presented.

- The identification approach (adaptation framework) clearly outperforms the naive approach and the random approach in the speed feature.
- There is no clear evidence of the difference in performance between the approaches, regarding the torque indicator.
- The adaptation framework obtained better averages for the success rates, the mean speed, and torque output.
- For the context values where the identification was not correct, the balanced requirements have a higher chance of choosing a final solution that do not fail, when compared to the speed requirements.
- The identification results indicate that the relationship between the context values $\boldsymbol{\theta}$, and the features of locomotion $\boldsymbol{b}$, seems to follow an approximately linear curve.

7.3 Making the DARwIn-OP walk up ramps with different slopes

The adaptation framework (Chapter 6) was also tested with the setup for the DARwIn-OP tests from section 5.2. The objective is to adapt the locomotion to different, unknown, values of ramp slope. The identification process, and the choice of a final solution, were conducted with basis on the sensitivity dataset obtained in the experiments from Section 5.2.4. The method was compared to a naive and a random approach that were both described in the previous section.

7.3.1 Adaptation framework setup

The dataset $\mathcal{D}_\mathrm{s}$, resulting from the sensitivity analysis on the Pareto fronts of the optimization experiments, contains solutions consisting of the values of seven different parameters of a CPG based controller (see Appendix C) that was applied to the locomotion task of a DARwIn-OP robot model. These solutions result in locomotion behaviors that are measured as four different indicators: the mean speed of the robot, an average related to the torque output of all the robot's joints, the MSE of the trajectory, and the distance between the CoP and the support polygon of the robot's feet. The behaviors produced are dependent on the value of a context variable modeled for the process, which in this case is the slope of a ramp that must be walked on upwards. In the sensitivity analysis, every solution was tested for slope values ranging from 0 to 12 degrees, with 1 degree intervals, and including both extremes.

This setup was tested for the adaptation to context values ranging from 0 to 12 degrees, with 0.5 degrees intervals and including the extremes. The cutoff values used for the adaptation algorithm's selection of the identification solution (see Section 6.3.1) were $\mathbf{c}_s = 10$ and $\mathbf{c}_d = 0.7$. Like in the iCub experiments, two different performance requirements were used in these tests: one focused only on the mean speed, and a more balanced approach. The first one had a hierarchy of priorities $\mathbf{h}_p =$ (mean speed) and weight $\mathbf{w}_p = (1.0)$, while the other had $\mathbf{h}_p = (\tau_{mean}$, mean speed, trajectory error, CoP distance) and $\mathbf{w}_p = (0.8, 0.7, 0.5, 0.5)$.

The tests were also repeated for two different sets of $\mathcal{D}_s$. One was the complete set, containing all the solutions, and respective behaviors in different contexts, from the sensitivity analysis in Section 5.2.4. The other one only contained behaviors for the context values of 0, 2, 4, 6, 8, 10, 12 degrees of the ramp's slope. These will be referred to as the complete and reduced sets, respectively. With the way the adaptation framework works, the identified value can only be one of the contexts tested for in these datasets.

The random and naive approaches tested for as points of comparison to the adaptation framework work in the same way as in the iCub experiments. Their descriptions can be found in Section 7.2.1.

7.3.2 Results

The results of the adaptation framework tests were compared against the other two approaches using Wilcoxon tests, similarly to the previous section. Averages of performance and the value estimated for each context in the identification phase are also shown, for both the complete and reduced sets of contexts from the sensitivity analysis.

Complete sensitivity dataset: Wilcoxon test results

Table 7.3 shows the direct comparison between the identification approach and the random and naive approaches, relative to the torque and speed features of the tests using the complete sensitivity dataset. The adaptation framework shows better values of mean speed than both approaches, for both sets of requirements tested. In three of these four cases, the p-value indicates a statistically significant difference. The identification approach's results related to the torque indicator are better in three cases, with one of them being significant.

Complete sensitivity dataset: performance averages

The performance values for each approach can give a better insight into these results. Table **??** shows the averages of some of these indicators for both performance requirements. The identification approach had the best success rates for both sets of performance requirements. In the case of the balanced requirements, it had 100% rate of trials without the robot falling, which drops to 48% when filtering for trials with a minimum speed of 0.0075 m/s. These trials, where the robot did not move at least

Table 7.3: Wilcoxon test results for our approach (labeled ID) against random and naive approaches, for the speed and torque features. Results related to the adaptation tests on the DARwIn-OP, using the complete sensitivity dataset. The values associated with a statistical significant test (p-value < 0.05) are bolded.

Performance requirements	Feature compared	Sum of the ranks of the differences		p-value
		ID approach	Random approach	
Balanced	Speed	**237**	33	**0.0076**
Balanced	Torque	87	183	0.1914
Speed	Speed	**276**	46	**0.0024**
Speed	Torque	96	226	0.0890
		ID approach	Naive approach	
Balanced	Speed	221	101	0.1209
Balanced	Torque	**37**	285	**0.0007**
Speed	Speed	**264**	58	**0.0051**
Speed	Torque	201	121	0.2355

15 cm, can be observed even in cases where the context value was identified correctly, which means the balanced performance requirements did not set a high enough weight in the speed indicator, resulting in solutions that fall short in that category. The tests with the speed focused requirements achieved a success rate of 68% for both definitions of the indicator. This indicates that the simulations that fall under the minimum speed are exclusively ones that result in a robot fall. These consisted in tests where the context value was misidentified, due to being a value not tested in the sensitivity analysis (*e.g* 0.5, 1.5, ...), meaning that the requirements focused on speed did not lead to the problem of not choosing a high enough speed. In addition to the success rates differences, the speed focused requirements also resulted in a higher average mean speed, considering all trials. On the other hand, the balanced requirements shows a lower average torque output.

Differences between the approaches and the performance requirements

Regarding the other two approaches tested, the random one results in increased performances above the speed threshold, when using the speed focused requirements. Given that there is a significant difference between the values for success rates with and without filtering in the balanced case, one can assume the increase comes from the previous hypothesis that the speed focused requirements are more efficient at selecting solutions that can move the robot an appreciable amount during locomotion. The naive approach does not show the same increase in performance with different requirements, but, in fact, a decrease (36% to 32%). This happens because the naive approach is highly dependent on how good the solution chosen by those requirements is in different

environments. Since it ignores the context variable, it chooses the same solution for each trial. In the case of the balanced requirements, it happens to be a solution that is slightly better at covering a minimum amount of ground, when compared to the speed focused requirements. In two cases the random and naive approach show better mean speeds when not considering robot falls, or speeds below the threshold. These results from the trials that do not fail tending to be ones related to higher speeds, which are also observable in the identification approach, but weighted down by the higher number of successful simulations. In a related situation, the naive approach shows the lowest average for the torque indicator for the speed requirements, but that is, again, tied to a much lower success rate (32% to 68%).

Complete sensitivity dataset: identification of context values

Figure 7.2 shows the value estimated in each for the context variable by the identification approach. Every context present in the sensitivity analysis (1, 2, ... 12) was correctly identified. For the rest of the values, given that they cannot be precisely estimated, the more desired outcome is a value directly below or above the nearest that can be estimated. This can be observed in the plot between 3 and 8 degrees, with an error of at most 1.5 degrees in the other cases. Given that the best overall success rate did not go over 68%, and that the failures observed were all in misidentified contexts, it is important to be able to correctly identify these values, or lessen the impact of these errors somehow.

Reduced sensitivity dataset

The reduced sensitivity dataset bases its identification process, and the choice of the final solution in a smaller group of contexts tested for each solution. Only the behaviors for the values of 0, 2, ... 12 were used, totaling seven different contexts, as opposed to the original twelve.

Reduced sensitivity dataset: Wilcoxon test results

The Wilcoxon test results from Table 7.4 indicate that the reduced dataset leads to a worse comparative performance than the complete set (Table 7.3), for every situation expect for the torque indicator performance in the balanced requirements, against the random approach. Here the sum of ranks of the differences is lower for the reduced dataset, leading to a p-value of approximately 0.0557, which is close to the threshold for it to be considered a statistically significant case. This might be a consequence of the adaptation framework being forced to identify context values which solutions are better at performing in certain environments, or it might result from the random approach being more hindered by the reduced set. The only situation where the identification approach comparative performance worsened enough to become significant was for the speed focused requirements, when comparing the torque indicator results against the naive approach. The naive approach, focusing on one solution through the

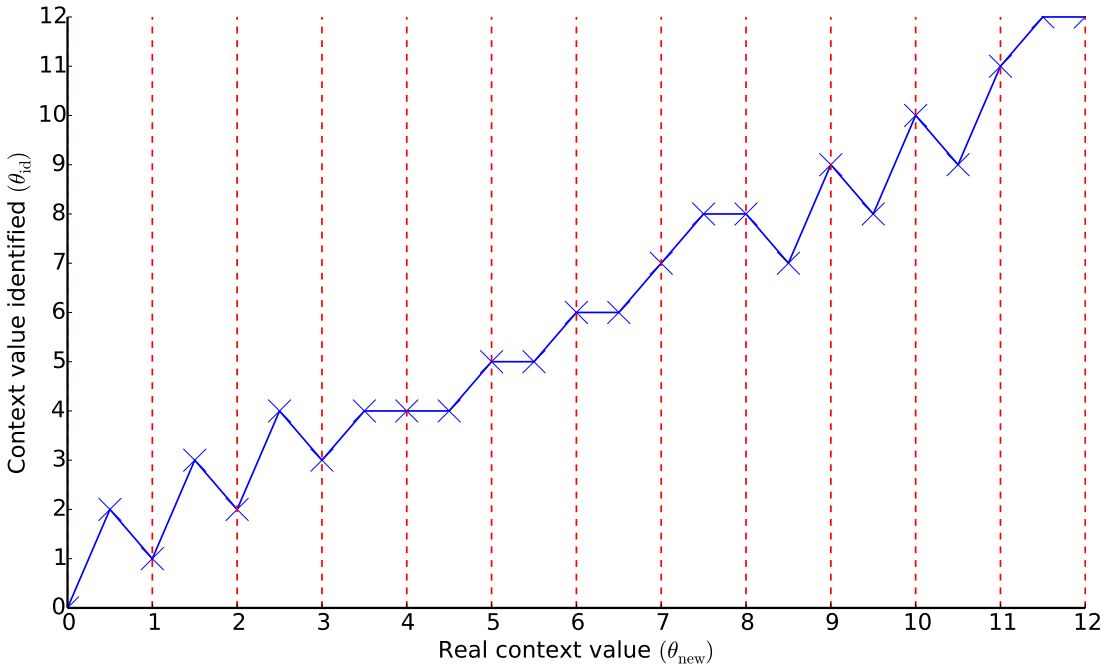


Figure 7.2: Real value of the context tested for, against the value identified by the adaptation framework. Results for the experiments with the DARwIn-OP on ramps with different slope values, using the complete sensitivity dataset. Vertical lines mark the context values which behaviors the solutions were tested for in the sensitivity analysis.

Table 7.4: Wilcoxon test results for our approach (labeled ID) against random and naive approaches, for the speed and torque features. Results related to the adaptation tests on the DARwIn-OP, using the reduced sensitivity dataset. The values associated with a statistical significant test (p-value < 0.05) are bolded.

Performance requirements	Feature compared	Sum of the ranks of the differences		p-value
		ID approach	Random approach	
Balanced	Speed	210	70	0.0562
Balanced	Torque	67	213	0.0557
Speed	Speed	225	90	0.0630
Speed	Torque	176	139	0.5663
		ID approach	Naive approach	
Balanced	Speed	93	220	0.1060
Balanced	Torque	**54**	261	**0.0071**
Speed	Speed	191	89	0.0879
Speed	Torque	228	**52**	**0.0072**

whole experiments, might have ended up choosing one that was comparatively better than before in multiple environments, or the identification approach might just have worsened enough. Both these situations can be cleared up when looking at the average performances of each approach in the different settings.

Reduced sensitivity dataset: performance averages

Table 7.5 shows the average performances for the reduced sensitivity test. These experiments led to worse results for the identification and random approaches, but to better ones in the balanced requirements of the naive approach. This approach showed the best success rate above the threshold speed, and the best averages for both speed indicators. On top of that, it had a better overall mean speed and success rates than the ones from the identification approach in the complete dataset tests. This highlights the impact of the balanced requirements in choosing a solution that successfully allows the robot to travel up a ramp. Even with the advantage of having the correct context information, in some cases, the choice falls in a solution that performs worse than the naive approach of ignoring the context variable.

Causes for some of the different performances, relative to the complete dataset

As previously stated, the performance of the naive approach is completely dependent on the solution chosen for every trial, which is the same. The one chosen in the reduced set led to better average performances. Using the speed focused requirements, the naive approach had worse success rates and better mean speeds, but its lower torque

Table 7.5: Success rates and average values of performance indicators for the adaptation phase tests for the DARwIn-OP, reduced sensitivity dataset. Shows results for all three approaches tested, with two different performance requirements. Success rates are shown both for strictly non falls, and filtered for a mean speed > 0.0075 m/s. Average mean speeds are shown both for all trials, and for trials with a mean speed > 0.0075 m/s. The best averages for each performance requirements are bolded.

		Success rate (%)		Average mean speed (m/s)		
Approach	Performance requirements	> 0.0075 m/s	Any speed	Overall	> 0.0075 m/s	Average $\boldsymbol{\tau}_{\text{mean}}(\text{N}^2.\text{m}^2)$
ID	Balanced	44.0	**100.0**	0.0444	0.1091	**6.26**
Random	Balanced	16.0	80.0	0.0063	0.1172	6.72
Naive	Balanced	**52.0**	52.0	**0.0664**	**0.1317**	6.76
ID	Speed	**60.0**	**60.0**	**0.0684**	0.1080	9.21
Random	Speed	32.0	36.0	0.0414	0.1146	8.66
Naive	Speed	28.0	28.0	0.0362	**0.1572**	**7.56**

output, in combination with the decreased performance for the identification approach, resulted in a statistically significant difference in the Wilcoxon test. The only relative improvement for the identification approach, seen in the torque indicator against the random approach, for the balanced requirements, seems to come essentially from a slightly worse average in the random approach, which was not enough to make the difference significant, although close (p-value = 0.0557).

Reduced sensitivity dataset: identification of context values

The values estimated for the context variable in each test for the reduced sensitivity dataset are presented in Figure 7.3. These are the same for both sets of performance requirements, since the identification step is independent of these. The errors observed in this plot are similar to those of Figure 7.2. In this situation, 2 degrees is the minimum step between the values in the reduced set (as opposed to 1 degree in the complete), meaning larger errors are expected. There are two outliers with a larger error, meaning that not only the reduced set has the unavoidable problem of not being able to identify as much different discreet values as before, but it also has a larger issue with identifying values close to those, resulting in greater errors, even when adjusting for the wider gap between each context value.

7.3.3 Conclusions

This section shows the main conclusions taken from the adaptation experiments related to the DARwIn-OP, which results are presented in Section 7.3.2.

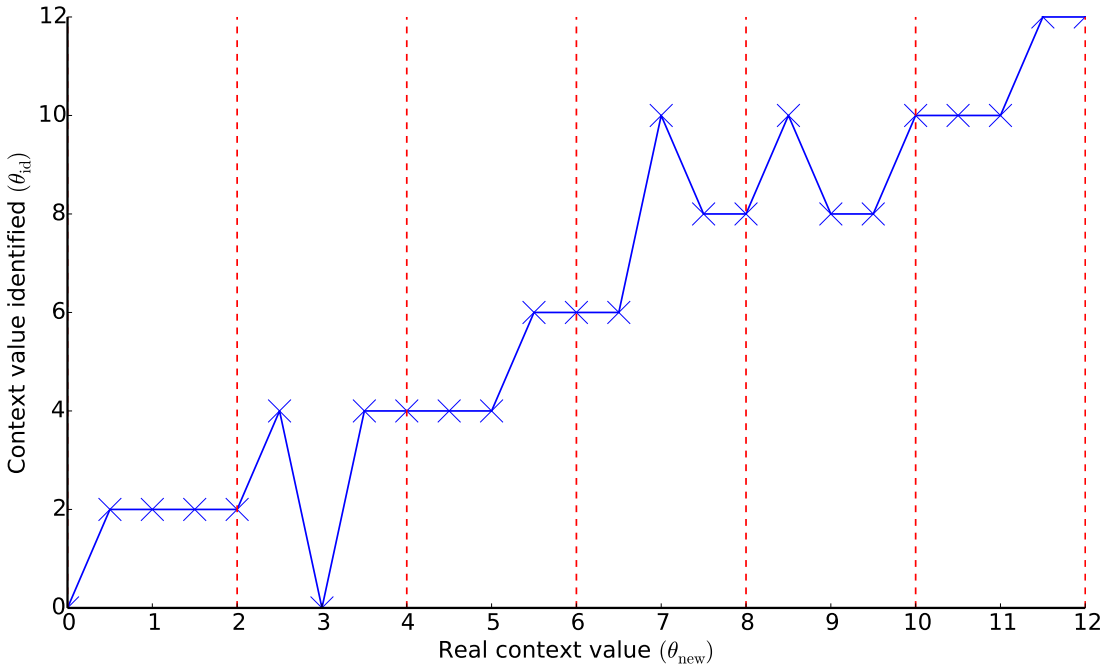


Figure 7.3: Real value of the context tested for, against the value identified by the adaptation framework. Results for the experiments with the DARwIn-OP on ramps with different slope values, using the complete sensitivity dataset. Vertical lines mark the context values which behaviors the solutions were tested for in the sensitivity analysis.

- Conclusions related to the complete sensitivity dataset:
  - The adaptation framework showed better performance values for both indicators. The differences in speed are clear, and the ones in the torque output are only significant in one instance.
  - The adaptation framework had lower average performances in some instances. However, these deficits are offset by a much larger success rate.
  - The weight set by the balanced requirements on the speed feature was not high enough to avoid situations where the robot moved less than 15 cm. The same did not happen with solutions chosen by the speed focused requirements.
  - As with the iCub experiments, the results of the identification of the context values suggest a relationship curve between these and the locomotion features that may be close to a line.
- Conclusions related to the reduced sensitivity dataset:
  - The adaptation framework showed worse relative performance when compared with the complete dataset, with no clear statistically significant overall advantages.
  - The identification approach shows worse results relative to the complete dataset. Specifically, the success rates do not offset the worse performance values, for the balanced requirements.
  - The good results for the naive approach highlight both the inability of the balanced requirement to choose solutions with an acceptable minimum speed, and the fact that the naive approach may get "stuck" with a good solution, regardless of the dataset size.

7.4 Discussion

This chapter presented the experiments related to the adaptation framework exposed in Chapter 6. Tests were conducted for both the iCub and the DARwIn-OP robots, trying to find solutions for their controllers that enabled them to walk on floors with different friction, or ramps with different slopes, respectively. Different performance requirements were tested for each one, prioritizing different locomotion features. The identification approach was tested against two others, called the random approach, and the naive approach.

Overall effect of the identification of the context value

The random approach and naive approach were both used to control for the possibility that the identification of the correct value for the environment variable was not advantageous for the adaptation process. Globally, the results show that a correct estimation

of this value brings a clear advantage, especially if the sensitivity dataset from where the necessary information is drawn is large enough. Even in some cases where the variable is estimated with a value that is close to the real value, but not equal to it, that advantage is still observed. There are, however, a lot of misidentified values, because of the fact that only some of them can be exactly estimated using the sensitivity dataset. Given that the misidentified tests are the main source of failed trials, this is an aspect which impact should be lessened somehow.

Results from the Wilcoxon tests

In the iCub Wilcoxon tests, the identification approach outperformed the other two in the speed feature, for both sets of requirements, with a statistically significant difference. The tests applied to the torque feature showed two significant cases, one that favored the identification approach, and other that favored the naive approach. The DARwIn-OP tests with the complete dataset showed the adaptation framework had a clearly better speed performance for three out of four cases, and one out of four cases for the torque indicator. The adaptation framework was not worse in a significant way for any of the comparisons. The reduced dataset led to comparatively worse performances, with only the torque indicator advantage from before being kept. Additionally, it led to the naive approach having a better torque output performance, when using the speed focused requirements.

Performance averages in the iCub experiments

These tests that rely on direct comparison can be supplemented by the performance averages of each approach in different categories. In the iCub tests, the identification approach showed better success rates, and better averages for both performance indicators, and for both sets of requirements, reinforcing the idea that the approach is well suited for this particular situation.

How different performance requirements affect different control architectures

The success rates shown in the DARwIn-OP's experiments are lower than the iCub ones, due to the nature of the environment variable; a ramp causes a physical obstacle that needs to be successfully traversed. The locomotion control frameworks used in both robots exacerbate this issue: the iCub was controlled by a dynamics based controller that has no trouble reaching an acceptable minimum displacement, while the DARwIn-OP was controlled by a CPG controller that does not guarantee forward movement. Another consequence of this is the balanced performance requirements resulting in worse success rates for this robot, since they do not prioritize speed enough to get the robot moving in some cases. The balanced requirements are still better at achieving a lower average torque output.

Performance averages in the DARwIn-OP experiments, complete sensitivity dataset

In the complete set results for the DARwIn-OP, the speed focused requirements showed a better success rate than the balanced requirements, which is the opposite of what was observed with the iCub. The random approach, which has an overall tendency of going up or down in performance that matches the identification approach, for all experiments, also has better performance with the speed focused requirements, unlike the naive approach.

Performance averages in the DARwIn-OP experiments, reduced sensitivity dataset

The tests conducted with the reduced sensitivity dataset led to worse average performances for the identification approach, which matches the drop observed in the Wilcoxon tests. The naive approach showed an improvement in the average performances in the balanced requirements tests. It also showed a higher success rate, and a better overall mean speed than the ID approach in the complete dataset, for the same requirements. This occurrence highlights both the problem with the balanced requirements with selecting effective solutions, even with a correct estimation of the context variable, and the possibility of the naive approach selecting an effective solution, which by chance is applicable to multiple contexts.

Advantages and drawbacks of a naive approach

The possibility of the naive approach ending up selecting a solution that performs well in a variety of contexts, also comes with the risk of selecting one that performs below average. For example, the iCub Wilcoxon tests show two statistically significant comparisons for the naive approach: one where it performs better, and other where it performs worse. The performance averages for the iCub experiments, as well as its success rates, are generally better for the random approach than those put out by the naive approach. In the reduced dataset for the DARwIn-OP, the naive approach had a better torque performance in the Wilcoxon test, using the balanced approach. This results from both the drop in performance of the identification approach in the reduced set, and the possibility of the naive approach to choose a good overall solution.

The hypothesis of a near linear relationship between the context variable and the locomotion features

When analyzing the individual identification of each environment, a hypothesis was put forward, stating that the same solution produces similar behaviors for similar values of the context variable. This means the intended behavior in the identification process is that, for the values that cannot be estimated correctly, the identification chooses values that are as close to the real value as possible. Both in the iCub's case, and in the complete dataset for the DARwIn-OP, zones approximating this behaviors were

identified. The reduced set for the DARwIn-OP showed a worse performance, even when taking into account that the wider gap between each context value in the set will naturally lead to larger errors.

Chapter 8

Summary and perspectives

For humanoid robots, locomotion is both an important and difficult to control task. Even finding control solutions that keep the robots standing can be difficult, and finding solutions that maximize performance is even harder. Locomotion controllers usually have open parameters that need to be tuned, and affect the outcome of the locomotion. This can be used to optimize the control towards features such as speed or energy consumption, or to adapt to changes to the terrain or the robot.

Broadly, approaches to humanoid locomotion control optimization can focus on an analysis of mathematical relations, or use a derivative-free approach. Optimization based on analysis is usually faster and potentially more precise, but they are not good at generalizing towards different controllers, or optimization objectives. Derivative free optimizations are better at generalizing, although they need take more time in offline optimizations, and usually do not guarantee the best performing solutions (only "good" ones).

The aim of the framework presented in this document is to offer an optimization and adaptation approach that works with any bipedal locomotion controller with open parameters. The optimization can be done towards different locomotion features, and adaptation to new terrains is done by using a context variable that models a change in the terrain/robot.

8.1 Goals

As a reminder, and for the sake of completeness of this summary, these are the main goals of the work derived from this thesis:

- **Optimize a humanoid locomotion controller towards multiple locomotion features, and different contexts.**
  - Locomotion features are defined as measurable quantities related to the task, such as speed, energy consumption, or stability measures.
  - Contexts define the combination of the robot model, the terrain it walks in, and the dynamics governing their interactions.

  - The optimizations setup should be as automated as possible, with minimal input from the users in terms of selection of parameters and their ranges, as well as stability features.
  - These optimizations can be conducted offline, and are not especially time constrained.

- **Adapt the locomotion to different contexts and user preferred features.**
  - The solution (set of parameters) chosen depends on both the identified context variable, and a hierarchy of user preferences for the locomotion features.
  - These adaptations should be conducted online, are time constrained, and should be as fast as possible.

8.2 Methodology and contributions

These are the methods used to achieve the proposed goals, and the contributions that results from each of them:

- **Run derivative-free optimizations that tune controller parameters towards multiple locomotion features.**
  - This can be applied to any controller with open parameters, which are used as optimization inputs.
  - Any locomotion feature measurable during the task, and represented as a single value, can be used as an optimization objective.

- **Run a preliminary optimization with a more extensive setup, and use a correlation analysis to chose the most relevant parameters and features for later optimizations**
  - This helps automates the process, by selecting the optimization parameters and objectives that most affect the chosen performance indicators.
  - The range of values for each parameter is also decided by leaving out the ones that always result in locomotion failures.

- **A context variable is created to model a terrain or robot change, and multiple optimizations are conducted, using different values of this variable.**
  - Any terrain or robot characteristic that can be measured, represented as a single value, and manipulated, can be used as the context variable.
  - This provides insight regarding how different solutions are required to control the locomotion in different contexts, and how these contexts impact the task's performance.

- **A sensitivity analysis is conducted by evaluating the best solutions for each optimization in multiple contexts.**
  - This will determine whether optimizing towards a single context is enough, by gauging if there is a solution adequate to all the contexts tested.
- **A procedure designed to adapt to an unknown terrain first tries to identify the value of the context variable, and then uses that information, together with the data from the previous optimizations, to choose a solution suited for the situation.** The identification is done by comparing the results of a trial run (in the unknown context) to the results of the previous optimizations. The final solution is chosen taking into account a hierarchy of user define preferences towards each locomotion feature (*e.g.*, prioritizing speed).
  - This adaptation procedure can be used to optimize the locomotion control in situations where certain context changes are expected.
  - The adaptation requires a trial run lasting a few seconds, and the computational cost is minimal, making it fast and suitable for online implementations.
  - The user preferences, being arranged in a hierarchy, and given different weights representing their importance, allow a choice between fast, or energy efficient and stable behaviors, for example.

8.3 Conclusions

Literature review

Serving as the introduction to the main topic of this thesis, and as literature review, Chapter 2 exposed the inherent complexity of the humanoid locomotion task, highlighting the difficulty to coordinate the different DoFs in a feasible gait, and making it stable. It also explored different approaches to the control of this task, in order to conclude how none of them is able to both optimize its open parameters towards clearly defined locomotion features, as well as adapt to changing terrain, while being applicable to different robots. It concluded that a derivative-free optimization approach, somehow abstract in the modeling of the context changes, would be the most adequate for the goals set in this project. The Inteligent Trial and Error algorithm from Cully et al. (2015) was identified as the reviewed work that more closely achieved these goals.

Optimization framework: exploration phase

Chapters 4 and 5 present the framework for the optimization of a given locomotion controller when applied to a given robot model, and the experiments conducted with it. The framework was designed to support any controller with open parameters, as long as it can be used to control the locomotion of the chosen humanoid robot model. In

order to highlight this point, the framework was tested in three different robot models (the iCub, the DARwIn-OP, and a virtual manikin), and two different controllers (a dynamics based controller with tasks hierarchy, and a Central Pattern Generators based controller).

The optimizations managed to reach higher speeds than those found in literature for both the iCub and the DARwIn-OP, and obtain sets of solutions that show diverse locomotion behavior in the three experimental setups. Adding stability measures as optimization objectives in the iCub experiments resulted in higher maximum speeds and a reduction if locomotion failures, at the cost of higher torque outputs. The correlation analysis conducted in the DARwIn-OP experiments was used to tune the initial optimization setup towards a less expensive one, both in terms of time and computationally. The new setup also resulted in improved maximum speeds.

These tests were also conducted in terrains with one varying parameter, and a sensitivity analysis showed that in no case there was a solution that could be optimal in every terrain tested, underlining the need for the adaptation to these changes.

Optimization framework: adaptation phase

Chapters 6 and 7 delineate the second stage of the thesis's method — the adaptation phase — and the experiments and respective results related to it. This framework supports itself with the data from the exploration phase, which lead to using two of the same robot models from the previous experiments (the iCub and the DARwIn-OP), and the same controllers.

The results show that, although there is not a suitable comparison with other works in literature, the correct identification of the terrain parameter improves the final locomotion performance. In the iCub experiments, the identification approach outperformed both a naive one that ignored the context variable, and an approach that selected the value of this variable randomly. The same was observed for the DARwIn-OP experiments with the complete sensitivity dataset. These results were worse with a reduced dataset, showing the importance that the data from the exploration phase has in a correct identification of the context, and, consequently, a good choice of a final solution.

The proposed strategy to encode user requirements towards different locomotion features, such as a faster gait, or a more energy conservative one, also showed that it is suitable to select solutions more geared towards those requirements. The requirements focused on speed always resulted in better overall average speeds for the identification approach. The balanced requirements, not being as focused as the former, led to mixed situations. They lead to higher success rates on the iCub, but the ramp in the DARwIn-OP trials meant that a focus on speed was more effective at achieving an acceptable minimum speed.

8.4 Discussion

As proposed, the overall adaptation framework, which includes an exploration phase to support the adaptation, can optimize any locomotion controller with open parameters, applied to any bipedal robot. The only constraint to this is that the robot, the controller, and interactions between them and surrounding terrain are simulated in a computer. The framework can optimize towards different contexts, as long as those changes are modeled under a numerical variable. Different locomotion features can be used as optimization objectives, and multiple examples were proposed, both performance and stability related. Both these features, and the controller parameters being optimized, can be automatically selected from a larger list, using a correlation analysis.

The adaptation scheme achieves the two main goals set for this phase: 1) it is conducted in two locomotion trials, with minimal time between them, for additional computations; 2) this adaptation can take into account a context variable included in the exploration phase, on top of user preferences towards different locomotion features.

8.4.1 Main advantages and disadvantages

The experiments with multiple setups showed the proposed framework can be applied to different humanoid robots, locomotion controllers, and context changes. It can also be applied to different situations, such as changing contexts and user preferences towards performance indicators. Implementation of the framework in these different setups is made easier with a correlation analysis that can be used to select the most relevant controller parameters and locomotion features. Additionally, the exploration phase can provide information for the design phase of both a robot (using design parameters in the optimization), and a controller, since it provides information about the potential locomotion behaviors they produce.

Making the framework flexible makes it less effective in specific situations. If the goal is to adapt a specific humanoid robot, which locomotion is controlled by a specific system, a more focused approach is more appropriate. As an example, making full use of the robot's sensors and their feedback, and adapting the controller to use that feedback, is usually a better approach for specific adaptations. Additionally, although the adaptation framework was developed with making everything require as little outside input as possible, the context parameter still needs to be specifically modeled in, with possible exceptions for common changes (such as terrain slope, friction, or height differences).

The length of the exploration phase can be a problem when adapting to a new type of context change. The impact of this problem is lessened by the fact that this time cost is predictable, as part of the design of the adaptation system, and in line with the time spent by other approaches in this implementation phases. On top of this, a shorter exploration phase can be employed by using previously obtained sets of optimal solutions, and only exploring those in the new environments.

Another point of concern is the possibility of the robot falling during the adaptation process. Even though the solution used for the identification is selected with safety in

mind, this is impossible to guarantee. The robot can be secured with some sort of safety device, but this can affect the locomotion process itself, changing the outcome of the desired solution. This outcome is already, in part, unpredictable, because of the issue of controllers evolved in simulation becoming less efficient when applied to a physical model of the robot. Koos, Mouret, and Doncieux (2013) proposed a solution to this problem, in the context of evolutionary algorithms, which is to model the concept of transferability as an optimization objective. They define this as simulation-to-reality disparity measure, and applied the framework to navigation tasks of an e-puck robot and a 8 DoF quadrupedal, finding efficient controllers for both cases.

8.4.2 Towards a full-fledged implementation of the adaptation framework

With such a big focus on usability and broad application potential, an important point to make is how far this framework is from being implementable as an actual application. Generally speaking, each optimization needs a framework that simulates the controller, the locomotion dynamics, the terrain, and a virtual robot model. This framework needs to be able to receive controller parameters to use in locomotion trials, and output features of locomotion after these end. Additionally, it needs to be able to control parts of the environment, to use as context variables.

Regarding the setup of the optimizations for the exploration phase, the implementation of locomotion features and context variables both come into question, since the parameters are dependent on the controller. In both cases, these measurements and variables need to be implemented in the simulation framework, case by case.

The features should be well defined, and common to the largest amount of humanoids possible. Features dependent on sensors should be avoided, with possible exceptions made for more common ones. These guidelines are meant to provide a list of features that can then be filtered with the results from a correlation analysis, but other features can be implemented by the user, provided that they can be measured in the simulation environment.

Similar to the locomotion features, a list of pre-defined context variables can also be developed, and more can be implemented, as long as they can be represented by a single value. Terrain changes such as friction and slopes levels can be easily modeled as a single real value, as long as they are implemented in the simulation. Variables that need the control of parts of the context which is not possible in some simulations can be implemented in other situations.

The adaptation phase, when applied to a physical robot, needs a controller implemented in the model, or a connection to a computer that can send commands to its joints. Measurements of locomotion features must be made either by reading the outputs of the robot's sensors, or by measuring these quantities with other devices. The adaptation framework needs to communicate with the robot, ideally through a link to a computer, since implementing the framework directly in the robot could require additional development time.

8.5 Perspectives

This sections presents overall thoughts on this work, as well as perspectives regarding future improvements.

8.5.1 Time cost and safety of the adaptation framework

One of the goals of this project was to keep the time spent in the adaptation process to a minimum. This needs to be balanced with the fact that, ideally, the process is safe at all time, and ensures the robot does not fall (or minimizes this possibility).

The exploration phase, requiring thousands of optimizations in different contexts, can take weeks. This phase is intended to be performed offline, and in a simulation environment. Having an exploration unconstrained by time, and by the computational limitations of a bipedal robot, can reduce the adaptation time.

The adaptation phase requires at least two trials: one to identify the solution, and another to evaluate if the performance resulting from the selected solution is acceptable. If feedback is introduced in the adaptation process for situations where the initially chosen solution does not provide a good performance, the process could go on for 3, 4, or more trials. In the experiments conducted in this work, each trial takes around 20 seconds to complete. Allowing for 10 seconds for setting up the robot between trials, 1 minute could be enough for the whole process, whereas an adaptation requiring 4 trials could take 2 minutes. This is an acceptable amount of time, but it could be reduced if the trials' duration was adaptive (see subsection 8.5.4).

Regarding safety, the adaptation process tries to keep robot falls to a minimum by choosing solutions safe in the largest amount of contexts possible. For the choice of the final solution, stability features can be given relatively high preferences in order to avoid locomotion failure. The success rates from the adaptation experiments in Chapter 7 show that different requirements are better at reaching safer solutions in different situations (balanced for the iCub in different CoFs, speed focused for the DARwIn-OP going up a ramp). Absolute safety is impossible to guarantee, since it depends on how risky the unknown context being adapted to is, and how much stability the locomotion control and robot provides.

8.5.2 Adding feedback to the adaptation process

The current framework has no feedback after an unsatisfactory adaptation. If the final solution results in a robot fall, or low values of performance, that information can be used to inform either a new identification, a new selection of the final solution, or both.

The identification process can be repeated with a different balance between the safety and the feature variability the solution provides. This can lead to a different identification, and, consequently, a different choice in the final stage. Another option is to, instead of re-doing the identification trial, directly changing the identified context value to one immediately above or below it (from the list of the values tested in the sensitivity analysis).

If the first identification is deemed to not be the source of the problem, or if the second one identifies the same context value, changes can be made to the choice of the final solution instead. If the issue is a locomotion failure, the set of user preferences can be modified to give a bigger weight to the stability features. If it is a performance issue, they can be made to focus that, or those, particular performance features instead.

8.5.3 Automatization of the exploration phase

The exploration phase requires the definition of what controller parameters need to be optimized, the locomotion features used as optimization objectives, and the context parameters modeled. This process can be time consuming and require user expertise. Making the exploration phase as automated as possible, by avoiding this necessities, accentuates the universal applicability of this approach.

The correlation analysis on a preliminary optimization can be used to filter down the list of parameters. Stability features, to be optimized along the performance features, can also be decided upon in a manner similar to the parameters, with the initial set consisting of measures that can be applied and measured to humanoid robots, regardless of the sensors they possess. Additionally, some measures dependent on common sensors can be considered (*e.g.*, using touch sensors on the feet).

The biggest obstacle towards the workflow evidenced in this project was the restrictions required by the DARwIn-OP in order to ensure it moved past the ramp, and not backwards, or not at all. The solution was to formulate the speed feature in a more focused way (specifying the direction), and adding a restriction to the minimum acceptable speed. This changes would not have affected the iCub optimizations, since the minimum speed was always observed (when it did not fall), and the LQP controller ensured the robot walked in the right direction. Taking this into account, it is possible that simply using this considerations in every situation would solve the problems where needed, and not negatively affect the rest of the optimizations.

Regarding the software implementation of the framework, it should be as independent as possible of the locomotion controller, and the simulation environment. The optimization algorithm decides the values of parameters to be optimized, and the framework sends that information to the simulator, receiving, in turn, the features resulting from the locomotion. It also should be able to tune the optimization setup with the information from the correlation analysis.

8.5.4 Duration of each trial

In the experiments conducted throughout this thesis, each individual trial took 20 seconds to complete. A possibility to minimize this duration, it to make it adaptive, ending the trial whenever the decision is made that the information gathered is enough.

If the locomotion features/optimization objectives used are an average over time, it is possible to track these values, and stop the simulation if they have changed over time in a similar manner, *i.e.*, if the locomotion shows periodicity, and that period is successfully detected.

8.5.5 Optimizing towards multiple context variables

One possible expansion to the optimization framework is expanding the context variable, $\boldsymbol{\theta}$, to a vector that contains multiple values, therefore modeling more than one aspect of the environment.

Given the number of simulations conducted for each value of $\boldsymbol{\theta}$ in the experiments in Chapter 5, there is a need to change the general approach, since adding more variables would result in, for instance, with two variabless, $\boldsymbol{n}_{\theta,1} \times \boldsymbol{n}_{\theta,2}$ optimizations, each one requiring thousands of simulations. $\boldsymbol{n}_{\theta,1}$ and $\boldsymbol{n}_{\theta,2}$ being the number of context values tested for each variable.

One way to reduce the number of optimizations would be optimizing towards one value of each context variable, and then testing those Pareto fronts in the $\boldsymbol{n}_{\theta}$ values, instead of conducting full optimizations. Choosing the values to do the initial optimizations would be a problem in itself. In the iCub experiments the sensitivity analysis showed the solutions optimized for the lowest value of coefficient of friction were the best at generalizing towards other values (see Section 5.1.5), while in the DARwIn-OP experiments these solutions were the ones resulting from the optimizations for values in the middle of the range tested (see Section 5.2.5).

For the adaptation phase, the choice of the solution for the identification process needs to be rethought. Ideally this solution would show diversity in the $\boldsymbol{n}_{\theta,1} \times \boldsymbol{n}_{\theta,2}$ contexts it was tested in. It could, however, be impossible to find a solution that does not show redundancy in terms of output for some of these combinations of values, due to the sheer number of these. An alternative would be to tend towards safer solutions in the final selection, protecting against a failed identification.

8.5.6 Relation between context values and locomotion features

The mathematical relationship between the values of the context variable $\boldsymbol{\theta}$ and the locomotion features $\boldsymbol{b}$ is important in the adaptation process, since the identification of the context depends on it.

Some evidence from the adaptation experiments in Chapter 7 supports the hypothesis that this relation may resemble a linear one, in some setups. A linear relationship, or simply one easy to encode and extrapolate, would mean that one could try to find this relationship from a small sample, and apply it to an expanded set of context values. This would open up the possibility of correctly identifying contexts not explored in the initial optimizations, as well as further informing the selection of the final adaptation solution.

8.5.7 Optimizing a humanoid for total mass and height

The optimizations in the XDE-manikin for different combinations of the robot's total mass and height (see Section 5.3) provided both expected, and somehow unexpected results. These types of optimizations can be important in the design phase of a bipedal robot. Other aspects of the robot, such as the length of each limb, or other parts of

its kinematic structure, as well as the mass of different sections, can be used in this context.

This hypothetical design phase optimization could be made more efficient by using the robot's mass and height as optimization inputs, along with controller parameters, or simply by themselves. This would make the optimization more focused in the design aspect, rather than finding a large Pareto front of solutions for one specific model of the robot.

An example of such an approach is the one presented by Maurice et al. (2017), which uses an evolutionary algorithm to optimize the design of a virtual humanoid. It encodes estimations of biomechanical demands during manual activities as 'ergonomic indicators", and optimizes parameters controlling the design of the robot. The most informative ergonomic indicators are used as optimization objectives.

Appendices

Appendix A

Description of the robot models used

This appendix describes the models of the robots used in this work: the iCub, the XDE-manikin, and the DARwIn-OP.

A.1 Description of the iCub model

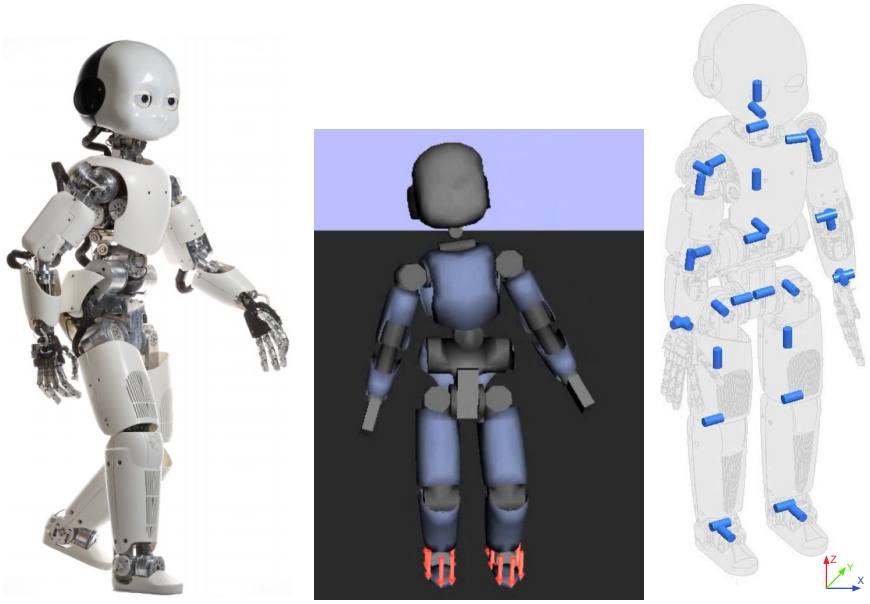
The iCub (Parmiggiani et al., 2012), seen in Figure A.1 (a) is approximately 1 m tall, weights approximately 24 kg, and has 53 DoFs. The virtual model used in the experiments in this thesis is a simplified version with not hand or eye joints, reduced to 32 joints, and with 4 contact points modeled for each foot.

The kinematic structure of the model (Figure A.1 (b)) used in the XDE simulator can be seen in Figure A.1 (c), and is described in Table A.1.

A.2 Description of the XDE-manikin

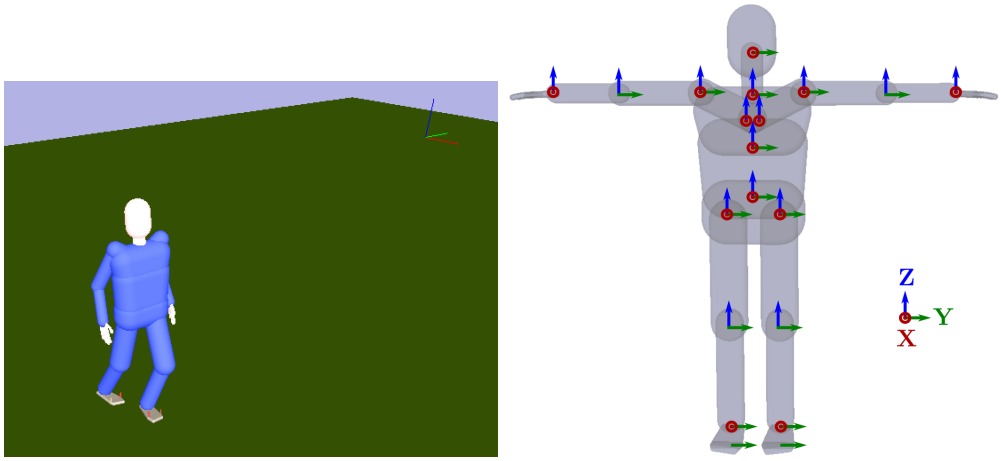
The XDE-manikin, displayed in Figure A.2 (a), is a virtual humanoid robot constructed for the XDE simulator (Merlhiot et al., 2012). When changing the height and mass of this system, its kinematic structure and the volumes of its body segments are maintained in the same proportions, although they are scaled with the changes, according to average scaling coefficients (*Open Design Lab Tools: Proportionality Constant Calculator*; *Mass of body segment - calculation*).

The robot has 21 segments and 20 joints, some of which have multiple DoFs, for a total of 45, not counting the 6 from the free floating base. This structure can be seen in Figure A.2 (b), and it is described in Table A.2.



(a) (b) (c)

Figure A.1: Images related to the iCub, showing a photograph of the robot (a) (from (Parmiggiani et al., 2012)), a screenshot of the model of the robot used in the XDE simulator (b), and the kinematic structure of the model used in XDE (c) (adapted from (Parmiggiani et al., 2012)).



(a) (b)

Figure A.2: Images related to the XDE-manikin, showing a screenshot of the robot in the XDE simulator (a), and the kinematic structure of the model (b) (from (Maurice, 2015)).

Table A.1: Kinematic structure and joint names of the model of the iCub robot. The coordinate system is the one used in the XDE simulations.

Body part	Joint	Axis
Head	Head roll	X
	Head pitch	Y
	Head yaw	Z
Torso	Torso roll	X
	Torso pitch	Y
	Torso yaw	Z
Right/Left arm	Shoulder roll	X
	Shoulder pitch	Y
	Shoulder yaw	Z
	Elbow pitch	Y
	Elbow yaw	Z
	Wrist roll	X
	Wrist pitch	Y
Right/Left leg	Hip roll	X
	Hip pitch	Y
	Hip yaw	Z
	Knee	Y
	Ankle roll	X
	Ankle yaw	Y

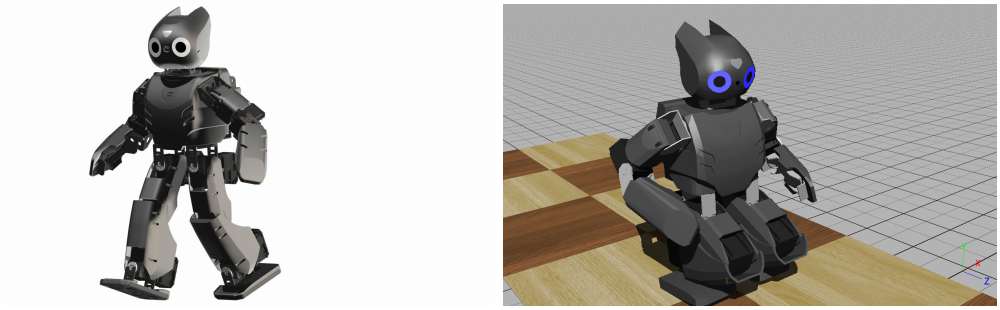
Table A.2: Kinematic structure, joint names, and number and direction of axes for each joint, for the XDE-manikin humanoid. Data from (Maurice, 2015)), using the coordinate system that was used in the XDE simulations.

Body part	Joint	Axes
Torso	Lower back	X Y Z
	Upper back	X Y Z
	Neck	X Y Z
	Head	X Y
Right/Left arm	Clavicle	X Z
	Shoulder	X Y Z
	Elbow	Y Z
	Wrist	X Z
Right/Left leg	Hip	X Y Z
	Knee	Y Z
	Ankle	X Y
	Toes	Y

A.3 Description of the DARwIn-OP model

The DARwIn-OP (Figure A.3 (a)) is an open source humanoid robot platform developed and manufactured by ROBOTIS in collaboration with the University of Pennsylvania (Ha et al., 2011). It is 0.455 m tall, weighs 2.8 kg, and has a total of 20 degrees of freedom.

The kinematic structure of the model (Figure A.3 (b) used in the Webots simulator can be seen in Figure A.4, and is described in Table A.3.



(a) (b)

Figure A.3: Images related to the DARwIn-OP, showing a photograph of the robot (a), and a screenshot of the model of the robot used in the Webots simulator (b).

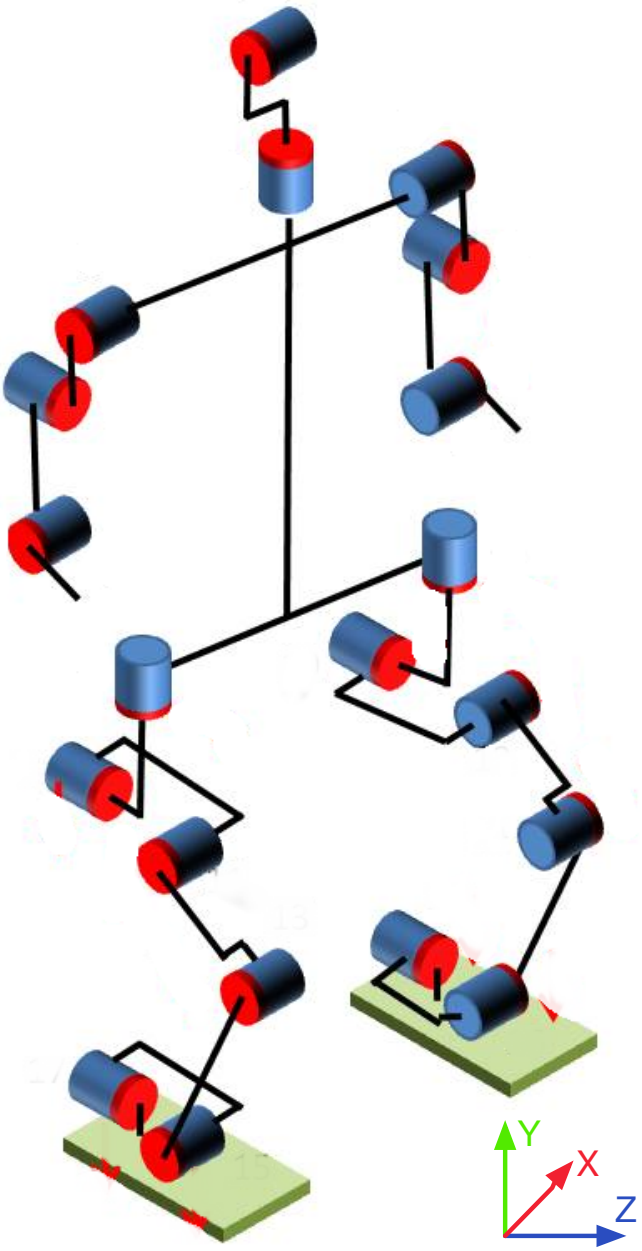


Figure A.4: Kinematic structure of the DARwIn-OP robot (adapted from (*RoboSavvy - Robots*)).

Table A.3: Kinematic structure and joint names of the model of the DARwIn-OP robot. The coordinate system is the one used in the Webots simulations, seen in Figure A.4.

Body part	Joint	Axis
Head	Head pitch	X
	Head yaw	Y
Right/Left arm	Shoulder roll	Z
	Shoulder pitch	X
	Elbow	X
Right/Left leg	Hip roll	Z
	Hip pitch	X
	Hip yaw	Y
	Knee	X
	Ankle roll	Z
	Ankle pitch	X

A.3.1 Webots simulation

The robot will be simulated in Webots (Michel, 2004), which is a platform developed by Cyberbotics Ltd and used to model, program and simulate mobile robots. Webots allows for a full dynamic simulation using a physics engine. It also provides the option for low level control of a robot's servos, and the simulation of the sensors present in the DARwIn-OP robot that are necessary to collect necessary feedback information. Furthermore, the code for its controllers can be written in C++, which is a flexible language with fast computation times that can also be used in the chosen robot. There is a open-source DARwIn-OP with Webots project (*Github: webots-cross-compilation*).

Webots also contains a model of DARwIn-OP out of the box. The simulated model of DARwIn-OP was designed to be as close as possible to the real robot. Is is equipped with the following sensors and actuators:

- 20 servos.
- 5 LED's (including 2 RGB ones).
- A 3 axes accelerometer.
- A 3 axes gyroscope.
- A camera.

A.3.2 DARwIn-OP physical limits

Each of the 20 Dynamixel MX-28 servos has the configuration presented in Table A.4.

Table A.4: DARwIn-OP's servos physical configuration.

Servo characteristic	Numeric limit	Units
maxForce	2.5	N.m
acceleration	55	rad/s^2
maxAngularVelocity	12.26	rad/s
dampingConstant	0.002	—
staticFriction	0.025	N.m

Appendix B

Dynamics based control with task hierarchy

This appendix briefly describes a dynamics based locomotion approach, where variables from dynamics equations are optimized through linear-quadratic programming (LQP) (Rockafellar, 1987). The control system also applies a task based hierarchy, where different sub-tasks are given relative importance based on weights. It was developed by Joseph Salini during his PhD thesis (Salini, 2012; Salini, Padois, and Bidaud, 2011), and implemented in C++ and used in XDE with python wrappers.

Additionally, we will detail, for each of the experiments done with this controller, the numbers used for the control variables that need to be tuned, but were not part of the optimization process.

B.1 Control scheme summary

Equation of motion and control objectives

The robot is considered a multi-body system modeled with rigid bodies, and applies bounded torque on each joint. It follows Euler-Lagrange motion equations,

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + N(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q}) + J_\chi(\boldsymbol{q})^\top \chi, \tag{B.1}$$

that relates the joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$, and accelerations $\ddot{\boldsymbol{q}}$, to the mass matrix $M(\boldsymbol{q})$, the nonlinear effects matrix $N(\boldsymbol{q}, \dot{\boldsymbol{q}})$, the gravity forces vector $\boldsymbol{g(q)}$ and the generalized wrench Jacobian $J_\chi(\boldsymbol{q})$, which describes how external forces affect the system in motion. χ is called the action variable, and is composed of the vector of contact forces, and the vector of torque inputs, $\chi = [\mathbf{w}_c^\top, \boldsymbol{\tau}^\top]$. $\mathbb{X}$ is called the dynamic variable of the system, and includes the joint accelerations, the contact forces, and the joint torques, $\mathbb{X} = [\ddot{\boldsymbol{q}}^\top, \chi^\top]$. The motion equation can be used to optimize the system towards any of these variables.

Tasks definition

This controller uses formalized tasks, defined as the servoing of a frame attached to the robot body to a desired goal value. A generic task function is defined as the norm of an error dependent on $\mathbb{X}$ which has to be minimized in the dynamic space

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbb{X}) = \| E(\boldsymbol{q})\mathbb{X} - f(\boldsymbol{q}, \dot{\boldsymbol{q}}) \|^2, \tag{B.2}$$

where $(E(\boldsymbol{q})\mathbb{X} - f(\boldsymbol{q}, \dot{\boldsymbol{q}}))$ is the error to minimize.

A task servoing controller is used to track and minimize the error of these task functions. For example, when tracking a twist,

$$\dot{\boldsymbol{t}}^{des} = \dot{\boldsymbol{t}}^{goal} + K_p \boldsymbol{\epsilon}_p + K_d \dot{\boldsymbol{\epsilon}}_p, \tag{B.3}$$

where $\dot{\boldsymbol{t}}^{goal}$ is the trajectory reference acceleration, ϵ_p, $\dot{\epsilon}_p$ are the pose and velocity errors, and K_p, K_d are the proportional and derivative gains reflecting respectively the stiffness and the damping of the virtual system.

The tasks used to achieve a desired high-level decision are subject to the equality and inequality constraints of the equations of motion and some actuation limits, as well as some incompatibilities between them, leading to the use of an optimization program, LQP. The actuation limits involve limiting the torque output of the system, and the range of motion and velocity of the actuators.

Frictional contact

The robot interacts with its environment through a set of contact points linked to its bodies. The i^{th} contact depends on two variables, one describing its velocity $\boldsymbol{v}_{c_i}$ and the other describing its force $\mathbf{w}_{c_i}$. Each one has a related Coulomb friction cone which relates the tangential component of the force with the normal component (Beer, Johnston, and Mazurek, 2013). A frictional contact has different discrete states, with two being distinguished here: the point does not move, or it takes off. When the contact is persistent the velocity is null and the forces lie inside the Coulomb cone, with the cone being approximated by a linear cone C_{c_i}. In the second case, the velocity along the contact normal $\boldsymbol{n}$ is greater than 0 and the the contact wrenches are null.

case 1 (contact is persistent): $\boldsymbol{v}_{c_i} = 0$

$$J_{c_i}(q)\ddot{\boldsymbol{q}} + \dot{J}_{c_i}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = 0 \tag{B.4}$$

$$C_{c_i}\mathbf{w}_{c_i} \leq 0 \tag{B.5}$$

case 2 (contact is lifting) : $v_{c_i}.n \geq 0$

$$\mathbf{w}_{c_i} = 0 \tag{B.6}$$

Table B.1: Tasks for the control of the iCub's and XDE-manikin's locomotion. The tasks can minimize the error of a frame or a joint. K_p and K_d are the proportional and derivative gains of the task tracking controllers.

Task	Type	w	$K_p(s^{-2})$	$K_d(s^{-1})$
Minimization of $\mathbb{X}$	-	$1e^{-7}$	0.01	$2\sqrt[2]{10}$
ZMP control	Frame	10	0	0
Right foot trajectory control	Frame	10	100	20
Left foot trajectory control	Frame	10	100	20
Contact points tasks	Frame	1	0	0
Posture task for the arms	Joint	0.01	9	6
Posture task for the leg	Joint	0.01	9	6
Posture task for the back	Joint	0.1	25	10
Task for the waist pose rotation	Frame	10	9	6
Task for the waist pose altitude	Frame	10	9	6
Contacts friction task	Frame	1	0	0

Tasks priorities: weighting strategy

A weighting strategy associates each task with a coefficient that sets its importance with respect to the others, the task weight $\boldsymbol{w}$.

B.2 Tasks and weights used for the locomotion control of the iCub and the XDE-manikin

The control framework was implemented in the iCub's and the XDE-manikin's locomotion control in the XDE simulator. The objective was to, at each time-step, solve a group of QPs to retrieve the torques for each joint. Let $\mathbb{X}_i^*$ be the solution to the problem:

$$\min \mathbb{X} \frac{1}{2}(\sum_{i=1}^{n}(w_i^2.T_i(q,\dot{q},\mathbb{X})) + w_0^2.T_0(q,\dot{q},\mathbb{X}))$$
$$s.t. : \begin{matrix} G\mathbb{X} \leq h \\ A\mathbb{X} = b \end{matrix}$$

The QP solver used was qld, with a sampling time of both XDE physics and the LQP solver of $0.01s$.

The tasks implemented in the controller for the iCub's locomotion are described in Table B.1. The feet trajectories tasks control the feet frames accelerations toward given trajectories derived from the ZMP trajectory calculated from the ZMP control system.

Contact tasks are used for each contact modeled, and are represented in the solver as an inequality constrain that represents the limitation of the contact force that must remain inside the cone of friction. The posture tasks maintain some of the joints at default values, so the robot stands in a normal position if the there are no walking controller commands. The waist pose tasks are used to keep a default "upright" pose of the robot. Additionally, feet and waist controllers that use a ZMP trajectory to generate feet and waist trajectories to be followed by their respective tasks. Regarding the actuation limits, the torque in all the joints is limited to 80 N.m, and no limits were imposed on the joints positions and velocities. In addition, a frictional contact constraint takes into account the relation between the tangential and vertical components of force acting on the feet so the robot doesn't fall. This ratio is constrained to a value of 1.5.

The contacts are added as tasks because the constraint of having a 0 m/s velocity when touching the ground can be treated either as a rigid equality constraint, or as an objective, in which case the acceleration of the contacts is minimized. This is more flexible and prevents the controller from finding no solutions in the case where stopping the motion completely is not feasible. Adding a contact task adds a contact point in the model of the contact contained in the XDE model instance, and it adds in the solver an inequality constraint that represents the limitation of the contact force that must remain inside the cone of friction.

B.3 Actuation limits constraints

The characteristics of the actuators bound their range of action. The torque is bounded in a way dependent on the dynamic variable of the system, χ,

$$\boldsymbol{\tau}_{min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max} \tag{B.7}$$

The range of motion and the velocity of the actuators are also bounded, and, in order to describe these in terms of χ, the following constraints are used,

$$\boldsymbol{q}_{min} \leq q + \dot{\boldsymbol{q}} h_1 + \ddot{\boldsymbol{q}}(h_1)^2/2 \leq \boldsymbol{q}_{max} \tag{B.8}$$

$$\dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}} + \ddot{\boldsymbol{q}} h_2 \leq \dot{\boldsymbol{q}}_{max} \tag{B.9}$$

where h_1 and h_2 are anticipation coefficients set to predict the future value of the state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ given the generalized acceleration $\ddot{\boldsymbol{q}}$ of the system.

In the iCub and XDE-manikin presented in this document, the maximum torque was set at 80 N.m. The joint position constraints were disabled, because they caused XDE to crash often, and no joint velocity and acceleration constraints were used.

Appendix C

CPG-based control

This appendix briefly describes a CPG based locomotion approach (Matos and Santos, 2012; Matos, 2013), that aims to be a model-free biped locomotion control approach allowing for easy parametrization, progressive building of a motor repertoire, and the inclusion of feedback mechanisms for modulation and adaptation of the generated joint trajectories. It was implemented in C++ in the Webots simulator, and used in simulations with a model of the DARwIn-OP in this project.

C.1 Control scheme summary

Each CPG produces the motions for a single leg, and is modeled in two layers: a rhythmic generation layer producing the temporal reference that feeds into a pattern generation layer to produce the spatial references. These references are represented as sums of motion primitives, which are encoded as a set of non-linear dynamical equations with well-defined attractor dynamics, and are smoothly regulated in regard to their amplitudes, frequencies, and pattern offsets. These motion primitives consist of sinusoidal and bell-shaped trajectories.

The same motions are produced for each leg, maintaining a contralateral anti-phase relationship. In order to simplify the walking behavior, it is assumed the feet are maintained parallel to the ground during the whole step cycle. Each motion pattern generator contains a set of motion primitives that can be disabled or modulated through parameter manipulation.

C.2 Rhythm generator

The rhythm generator layer is implemented as a coupled phase oscillator,

$$\dot{\boldsymbol{\phi}}_i = \boldsymbol{\omega} + \boldsymbol{k} \sin(\boldsymbol{\phi}_i - \boldsymbol{\phi}_o + \pi), \tag{C.1}$$

which results in $\boldsymbol{\phi}_i$ being an increasing periodic signal that is used as the phase of the leg i, with rate $\boldsymbol{\omega}$. $\boldsymbol{\phi}_o$ is the phase of leg o, kept in a desired relationship with the

oscillator for i. The coupling strength can be controlled with k.

C.3 Motion primitives

A motion pattern generator, implemented as a set of nonlinear dynamical equations, generates a list of positions for each joint, taking into account the primitives for all the desired motions. The position $\boldsymbol{z}_{j,i}$ of joint j from leg i is generated according to the current phase of the leg, $\boldsymbol{\phi}_i$, obtained through C.1,

$$\dot{\boldsymbol{z}}_{j,i} = \boldsymbol{\alpha}(O_{j,i} - \boldsymbol{z}_{j,i}) + \sum f(\boldsymbol{z}_{j,i}, \boldsymbol{\phi}_i, \dot{\boldsymbol{\phi}}_i). \tag{C.2}$$

The offset attractor $O_{j,i}$ is a position $\dot{\boldsymbol{z}}_{j,i}$ converges to if $\boldsymbol{\alpha} > 0$. Each motion primitive is defined by a function $f(\boldsymbol{z}_{j,i}, \boldsymbol{\phi}_i, \dot{\boldsymbol{\phi}}_i)$.

Each motion function can be the form of either a sinusoidal profile,

$$f_j^{\text{motion}} = -A_{j,\text{motion}} \dot{\boldsymbol{\phi}}_i \sin(\boldsymbol{\phi}_i + \boldsymbol{\psi}_{\text{motion}}), \tag{C.3}$$

or a bell-shaped profile,

$$f_j^{\text{motion}} = \frac{A_{j,\text{motion}} \dot{\boldsymbol{\phi}}_i (\boldsymbol{\phi}_i + \boldsymbol{\psi}_{\text{motion}})}{\boldsymbol{\sigma}^2} \exp\Big(- \frac{(\boldsymbol{\phi}_i + \boldsymbol{\psi}_{\text{motion}})^2}{2\boldsymbol{\sigma}^2} \Big). \tag{C.4}$$

In these equations, A_{motion} is the amplitude of the motion, ϕ the phase of the current leg, ψ_{motion} the phase shift, and $\boldsymbol{\sigma}$ the width of the bell curve.

Next, a set of motion primitives put together to achieve basic goal-directed bipedal walking is presented. These equations resort only to sinusoidal and bell-shaped joint trajectory profiles.

Balancing Motion

The balancing motion displaces the CoM in the frontal plane from one foot to the other, in order to place the hold on the current support foot. It acts on hip and ankle rolls as a sinusoidal trajectory that makes the robot oscillate laterally. If the motion of these joints is symmetrical, the feet will be parallel to the ground.

Flexion Motion

The flexion motion is meant to achieve the vertical clearance of the foot in the swing phase. It changes the vertical length of the leg by actuating in the three pitch joints: hip, knee, and ankle. The hip and knee trajectories are bell-shaped curves, and the motion in the ankle is the sum of the previous two, imposing a parallel foot to the ground.

Compass Motion

The compass motion is responsible for the propulsion of the body during locomotion. It moves the legs in the sagittal plane, alternating between the contralateral legs forward and backward. The motions are described as sinusoidal curves for the hip and ankle pitch joints.

Knee Yielding Motion

The knee yielding motion is used in the supporting foot at the start of the stance phase. This is done when the body weight is passed to the foot, and it flattens the vertical trajectory of the CoM. The joint trajectories are applied to the knee and ankle joints, and are described by a sinusoidal profile.

Pelvis Rotation Motion

Alternate rotation of the pelvis can contribute to improve stability by smoothing the inflections when changing the vertical direction of the CoM. This motion can also increase the horizontal length of a step by twisting the body and placing the swinging foot further in front. This pelvic rotation is implemented as a sinusoidal trajectory applied to the hip yaw joints.

C.3.1 Motion primitives parametrization

Selecting different amplitudes and offsets for the various motion primitives described above changes some features of the walking task.

Changing the offset values will change the initial posture of the robot. This posture usually consists of flexed legs with a slight forward tilt, allowing for the onset of a swing phase. Offsets of the hip and ankle roll joints are set as symmetrical to make the legs point inward in the frontal plane. Offsets for the hip, knee and ankle pitch joints define the initial vertical leg length.

The amplitude of the balancing motion affects the movement transferring the CoM over the two feet, alternately. An amplitude that makes the displacement of the body weight get directly above the center of the support feet is probably preferred. Going past that value might make the CoP go over the valid support region. Changing the amplitude of the flexion motion changes the total vertical clearance of the feet in the swing phase. The amplitude of the compass motion is directly tied to the length of each step (thesis mentions an almost linear relation).

The influence of each motion primitive parameter in the final trajectory of the robot is not clear. When they are combined, the achieved kinematic and dynamic behaviors are difficult to anticipate.

C.3.2 List of the CPG controller's parameters tuned

Here are listed all the parameters tuned in the optimization process. In addition to these, other, lower level parameters, could possibly be used, but they have a lower impact in the behavior of the locomotion, and a list of twenty parameters is already one that needs to be decreased for this kind of optimization to work effectively.

- $\boldsymbol{\alpha}$: Time constant of the dynamical equations for the motion primitives. Higher values increase the rate of change of the outputs. A value of $\alpha > 0$ is required for the stability of the dynamical system.
- $\boldsymbol{T}$: Period of oscillation of the generators oscillators of both legs.
- $\mathbf{A_{balancing,hip}}$: Amplitude for the hip roll function of the balancing motion.
- $\mathbf{A_{balancing,ankle}}$: Amplitude for the ankle roll function of the balancing motion.
- $\mathbf{A_{flexion,hip}}$: Amplitude for the hip pitch function of the flexion motion.
- $\mathbf{A_{flexion,knee}}$: Amplitude for the knee pitch function of the flexion motion.
- $\mathbf{A_{flexion,ankle,h}}$: Amplitude for the ankle pitch function of the flexion motion, hip function contribution.
- $\mathbf{A_{flexion,ankle,k}}$: Amplitude for the ankle pitch function of the flexion motion, knee function contribution.
- $\mathbf{A_{compass,hip}}$: Amplitude for the hip pitch function of the compass motion.
- $\mathbf{A_{compass,ankle}}$: Amplitude for the ankle pitch function of the compass motion.
- $\mathbf{A_{yield,knee}}$: Amplitude for the knee pitch function of the knee yielding motion.
- $\mathbf{A_{yield,ankle}}$: Amplitude for the ankle pitch function of the knee yielding motion.
- $\boldsymbol{\sigma}_{\mathbf{flexion,hip}}$: Amplitude of the bell curve for the flexion motion for the hip, *i.e.* the duration of the movement.
- $\boldsymbol{\sigma}_{\mathbf{flexion,knee}}$: Amplitude of the bell curve for the flexion motion for the knee, *i.e.* the duration of the movement.
- $\mathbf{O_{hip,pitch}}$: Offset for the hip pitch trajectory, left and right equal.
- $\mathbf{O_{knee,pitch}}$: Offset for the knee pitch trajectory, left and right equal.
- $\mathbf{O_{ankle,pitch}}$: Offset for the ankle pitch trajectory, left and right equal.
- $\mathbf{O_{hip,roll}}$: Offset for the hip roll trajectory, left and right symmetrical.
- $\mathbf{O_{ankle,roll}}$: Offset for the ankle roll trajectory, left and right symmetrical.
- $\mathbf{O_{hip,yaw}}$: Offset for the hip yaw trajectory, left and right symmetrical.

Bibliography

Adekanmbi, Oluwole and Paul Green (2015). "Conceptual comparison of population based metaheuristics for engineering problems." In: *The Scientific World Journal* 2015, p. 936106 (cit. on p. 32).

Afshar, Parsa Nassiri and Lei Ren (2012). "Dynamic Stability of Passive Bipedal Walking on Rough Terrain: A Preliminary Simulation Study". In: *Journal of Bionic Engineering* 9.4, pp. 423–433 (cit. on p. 2).

Alexander, R. McN. (1984). "The Gaits of Bipedal and Quadrupedal Animals". In: *The International Journal of Robotics Research* 3.2, pp. 49–59 (cit. on p. 9).

Alexander, R. McNeill. (1996). *Optima for animals*. Princeton University Press, p. 169 (cit. on p. 7).

Amaran, Satyajith et al. (2016). "Simulation optimization: a review of algorithms and applications". In: *Annals of Operations Research* 240.1, pp. 351–380 (cit. on p. 32).

Aoi, Shinya and Kazuo Tsuchiya (2005). "Locomotion Control of a Biped Robot Using Nonlinear Oscillators". In: *Autonomous Robots* 19.3, pp. 219–232 (cit. on pp. 17, 20, 27).

— (2011). "Generation of bipedal walking through interactions among the robot dynamics, the oscillator dynamics, and the environment: Stability characteristics of a five-link planar biped robot". In: *Autonomous Robots* 30.2, pp. 123–141 (cit. on pp. 18, 27).

Asif, Umar and Javaid Iqbal (2011). "An Approach to Stable Walking over Uneven Terrain Using a Reflex-Based Adaptive Gait". In: *Journal of Control Science and Engineering* 2011, pp. 1–12 (cit. on p. 47).

B, Amir Massah et al. (2012). "An Open Loop Walking on Different Slopes for NAO Humanoid Robot". In: *Procedia Engineering* 41, pp. 296–304 (cit. on p. 47).

Beardmore, Roy. *Coefficients Of Friction*. URL: http://www.roymech.co.uk/Useful_Tables/Tribology/co_of_frict.htm#method (visited on 10/10/2017) (cit. on p. 69).

Beer, Ferdinand P. (Ferdinand Pierre), E. Russell (Elwood Russell) Johnston, and David F. (David Francis) Mazurek (2013). *Vector mechanics for engineers. Statics*. McGraw-Hill (cit. on pp. 67, 157).

Beyer, H.-G. and K. Deb (2001). "On self-adaptive features in real-parameter evolutionary algorithms". In: *IEEE Transactions on Evolutionary Computation* 5.3, pp. 250–270 (cit. on p. 37).

Blickle, Tobias and Lothar Thiele (1996). "A Comparison of Selection Schemes Used in Evolutionary Algorithms". In: *Evolutionary Computation* 4.4, pp. 361–394 (cit. on p. 37).

Brogliato, Bernard (1996). *Nonsmooth Impact Mechanics*. Vol. 220. Lecture Notes in Control and Information Sciences. London: Springer-Verlag (cit. on p. 1).

Brownlee, Jason. (2011). *Clever algorithms : nature-inspired programming recipes*. Lulu, p. 423 (cit. on p. 40).

Calandra, Roberto et al. (2016). "Bayesian optimization for learning gaits under uncertainty". In: *Annals of Mathematics and Artificial Intelligence* 76.1-2, pp. 5–23 (cit. on p. 22).

Chang, Yuqing, Zyed Bouzarkouna, and Deepak Devegowda (2015). "Multi-objective optimization for rapid and robust optimal oilfield development under geological uncertainty". In: *Computational Geosciences* 19.4, pp. 933–950 (cit. on p. 38).

Clower, William T (1998). "Early Contributions to the Reflex Chain Hypothesis". In: *Journal of the History of the Neurosciences* 7.1, pp. 32–42 (cit. on p. 16).

Črepinšek, Matej, Shih-Hsi Liu, and Marjan Mernik (2013). "Exploration and exploitation in evolutionary algorithms". In: *ACM Computing Surveys* 45.3, pp. 1–33 (cit. on p. 59).

Cully, Antoine et al. (2015). "Robots that can adapt like animals". In: *Nature* 521.7553, pp. 503–507 (cit. on pp. 3, 26, 28, 29, 47, 138).

Dai, Cai, Yuping Wang, and Miao Ye (2015). "A new multi-objective particle swarm optimization algorithm based on decomposition". In: *Information Sciences* 325, pp. 541–557 (cit. on p. 51).

Dai, Hongkai and Russ Tedrake (2016). "Planning robust walking motion on uneven terrain via convex optimization". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 579–586 (cit. on p. 12).

Davidor, Yuval. (1991). *Genetic algorithms and robotics : a heuristic strategy for optimization*. World Scientific, p. 164 (cit. on p. 79).

Deb, K. et al. (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". English. In: *IEEE Transactions on Evolutionary Computation* 6.2, pp. 182–197 (cit. on pp. 36, 40–42, 51, 76).

Deb, Kalyanmoy. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, p. 497 (cit. on pp. 32, 36, 59).

Deb, Kalyanmoy, Manikanth Mohan, and Shikhar Mishra (2005). "Evaluating the ϵ-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions". In: *Evolutionary Computation* 13.4, pp. 501–525 (cit. on p. 51).

Diedam, H. et al. (2008). "Online walking gait generation with adaptive foot positioning through Linear Model Predictive control". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1121–1126 (cit. on pp. 14, 26).

Dong, Hao, Mingguo Zhao, and Naiyao Zhang (2011). "High-speed and energy-efficient biped locomotion based on Virtual Slope Walking". In: *Autonomous Robots* 30.2, pp. 199–216 (cit. on p. 47).

Dorigo, M., V. Maniezzo, and A. Colorni (1996). "Ant system: optimization by a colony of cooperating agents". In: *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 26.1, pp. 29–41 (cit. on p. 33).

Eaton, Malachy (2015). "Evolutionary Algorithms and the Control of Systems". In: pp. 9–20 (cit. on p. 34).

Eiben, A. E. and J. E. Smith (2003). *Introduction to Evolutionary Computing*. Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on pp. 34, 35).

Fujiki, Soichiro et al. (2012). "Improving adaptive walking of a biped robot on a splitbelt treadmill by controlling the interlimb coordination". In: *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, pp. 396–401 (cit. on pp. 23, 29).

Gay, Sébastien (2014). "Visual Control of Legged Robots for Locomotion in Complex Environments using Central Pattern Generators". In: (cit. on pp. 2, 25, 28, 29).

Gen Endo et al. (2005). "Experimental Studies of a Neural Oscillator for Biped Locomotion with QRIO". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, pp. 596–602 (cit. on pp. 18, 27).

Github: webots-cross-compilation. URL: https://github.com/darwinop/webots-cross-compilation (visited on 03/17/2017) (cit. on p. 153).

Glover, Fred. and Gary A. Kochenberger (2003). *Handbook of metaheuristics*. Kluwer Academic Publishers, p. 556 (cit. on p. 33).

Gregg, Robert D, Yasin Dhaher, and Kevin M Lynch (2011). "Functional asymmetry in a five-link 3D bipedal walker." In: *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference* 2011, pp. 7820–3 (cit. on p. 49).

Gregg, Robert D et al. (2011). "The basic mechanics of bipedal walking lead to asymmetric behavior." In: *IEEE ... International Conference on Rehabilitation Robotics : [proceedings]* 2011, p. 5975459 (cit. on p. 49).

Grillner, Sten (2006). "Biological Pattern Generation: The Cellular and Computational Logic of Networks in Motion". In: *Neuron* 52.5, pp. 751–766 (cit. on p. 16).

— (2011). "Control of Locomotion in Bipeds, Tetrapods, and Fish". In: *Comprehensive Physiology*. Hoboken, NJ, USA: John Wiley & Sons, Inc. (cit. on p. 16).

Ha, Inyong, Yusuke Tamura, and Hajime Asama (2011). "Gait pattern generation and stabilization for humanoid robot based on coupled oscillators". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3207–3212 (cit. on p. 81).

Ha, Inyong et al. (2011). "Development of open humanoid platform DARwIn-OP". English. In: *Proc. of IEEE SICE 2011*. IEEE, pp. 2178–2181 (cit. on pp. 4, 151).

Handžić, Ismet and Kyle B Reed (2015). "Perception of gait patterns that deviate from normal and symmetric biped locomotion." In: *Frontiers in psychology* 6, p. 199 (cit. on p. 49).

Hansen, Nikolaus (2006). "The CMA evolution strategy: a comparing review." In: *Towards a New Evolutionary Computation*. Springer Berlin Heidelberg, pp. 75–102 (cit. on pp. 21, 51).

Hauser, Helmut et al. (2011). "Biologically inspired kinematic synergies enable linear balance control of a humanoid robot". In: *Biological Cybernetics* 104.4-5, pp. 235–249 (cit. on pp. 19, 27).

Hemker, T. et al. (2009). "Efficient Walking Speed Optimization of a Humanoid Robot". In: *The International Journal of Robotics Research* 28.2, pp. 303–314 (cit. on pp. 23, 29).

Herdt, A, N Perrin, and Pierre-Brice Wieber (2010). "Walking without thinking about it". English. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 190–195 (cit. on pp. 2, 14, 15, 26).

Herdt, Andrei et al. (2010). "Online Walking Motion Generation with Automatic Foot Step Placement". In: *Advanced Robotics* 24.5-6, pp. 719–737 (cit. on pp. 14, 15).

Hiratsuka, Michihisa et al. (2016). "Trajectory learning from human demonstrations via manifold mapping". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3935–3940 (cit. on p. 18).

Hobon, Mathieu, Nafissa Lakbakbi Elyaaqoubi, and Gabriel Abba (2013). "Influence of frictions on gait optimization of a biped robot with an anthropomorphic knee". In: *2013 9th Asian Control Conference (ASCC)*. IEEE, pp. 1–6 (cit. on p. 47).

Hodgkin, A L and A F Huxley (1952). "A quantitative description of membrane current and its application to conduction and excitation in nerve." In: *The Journal of physiology* 117.4, pp. 500–44 (cit. on p. 16).

Holmes, Philip et al. (2006). "The Dynamics of Legged Locomotion: Models, Analyses, and Challenges". In: *SIAM Review* 48.2, pp. 207–304 (cit. on p. 2).

Hopkins, Michael A., Dennis W. Hong, and Alexander Leonessa (2014). "Humanoid locomotion on uneven terrain using the time-varying divergent component of motion". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 266–272 (cit. on pp. 25, 28, 29).

Hu, Yue et al. (2016). "Walking of the iCub humanoid robot in different scenarios: Implementation and performance analysis". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 690–696 (cit. on p. 63).

Hurmuzlu, Yildirim, Frank Génot, and Bernard Brogliato (2001). *Modeling, Stability and Control of Biped Robots A General Framework*. en. Tech. rep. (cit. on p. 21).

— (2004). "Modeling, stability and control of biped robots—a general framework". In: *Automatica* 40.10, pp. 1647–1664 (cit. on p. 49).

Hyon, S. and T. Emura (2005). "Symmetric Walking Control: Invariance and Global Stability". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1443–1450 (cit. on p. 49).

Ibanez, Aurelien, Philippe Bidaud, and Vincent Padois (2014). "Automatic Optimal Biped Walking as a Mixed-Integer Quadratic Program". In: *Advances in Robot Kinematics*. Cham: Springer International Publishing, pp. 505–516 (cit. on p. 63).

Igel, Christian, Nikolaus Hansen, and Stefan Roth (2007). "Covariance Matrix Adaptation for Multi-objective Optimization". In: *Evolutionary Computation* 15.1, pp. 1–28 (cit. on p. 51).

Ijspeert, Auke Jan (2008). "Central Pattern Generators for Locomotion Control in Animals and Robots: A Review". In: *Neural Networks* 21, pp. 642–653 (cit. on pp. 17, 27).

Ijspeert, Auke Jan et al. (2013). "Dynamical movement primitives: learning attractor models for motor behaviors." In: *Neural computation* 25.2, pp. 328–73 (cit. on p. 18).

Jahn, Johannes (1985). "Scalarization in Multi Objective Optimization". In: *Mathematics of Multi Objective Optimization*. Vienna: Springer Vienna, pp. 45–88 (cit. on p. 35).

Jones, J. Gareth, E.M. (Tilli) Tansey, and Douglas G. Stuart (2011). "Thomas Graham Brown (1882–1965): Behind the Scenes at the Cardiff Institute of Physiology". In: *Journal of the History of the Neurosciences* 20.3, pp. 188–209 (cit. on p. 16).

Kajita, S. and K. Tani (1991). "Study of dynamic biped locomotion on rugged terrain-theory and basic experiment". In: *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*. IEEE, 741–746 vol.1 (cit. on p. 2).

Kajita, S et al. (2003). "Biped walking pattern generation by using preview control of zero-moment point". In: *2003 IEEE International Conference on Robotics and Automation Cat No03CH37422* 2.2, pp. 1620–1626 (cit. on pp. 13, 26).

Kajita, S. et al. (2004). "Biped walking on a low friction floor". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 4. IEEE, pp. 3546–3552 (cit. on p. 47).

Kajita, Shuuji et al. (2001). "The 3D Linear Inverted Pendulum Mode: A Simple Modeling for a Biped Walking Pattern Generation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* 1.4, pp. 239–246 (cit. on pp. 9, 13).

Kajita, Shuuji et al. (2014). *Introduction to Humanoid Robotics*. Vol. 101. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on pp. 7, 9).

Katz, P S (1996). "Neurons, networks, and motor behavior." In: *Neuron* 16.2, pp. 245–53 (cit. on p. 15).

Kennedy, J. and R. Eberhart (1995). "Particle swarm optimization". English. In: *Proc. of IEEE ICNN 1995*. Vol. 4. IEEE, pp. 1942–1948 (cit. on pp. 22, 34, 51).

Khachaturyan, A. et al. (1981). "The thermodynamic approach to the structure analysis of crystals". In: *Acta Crystallographica Section A* 37.5, pp. 742–754 (cit. on p. 33).

Kieboom, J. van den (2009). "Biped locomotion and stability : a practical approach". PhD thesis (cit. on pp. 22, 29).

Kiehn, Ole (2006). "Locomotor circuits in the mammalian spinal cord". In: *Annual Review of Neuroscience* 29.1, pp. 279–306 (cit. on p. 15).

Kimura, H., Y. Fukuoka, and A. H. Cohen (2007). "Adaptive Dynamic Walking of a Quadruped Robot on Natural Ground Based on Biological Concepts". In: *The International Journal of Robotics Research* 26.5, pp. 475–490 (cit. on pp. 2, 25, 29).

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). "Optimization by Simulated Annealing". In: *Science* 220.4598 (cit. on p. 33).

Kober, Jens and Jan Peters (2012). "Reinforcement Learning in Robotics: A Survey". In: Springer Berlin Heidelberg, pp. 579–610 (cit. on p. 23).

Koos, Sylvain, Jean-Baptiste Mouret, and Stéphane Doncieux (2013). "The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics". In: *IEEE Transactions on Evolutionary Computation* 17.1, pp. 122–145 (cit. on p. 141).

Kuo, Arthur D, J Maxwell Donelan, and Andy Ruina (2005). "Energetic consequences of walking like an inverted pendulum: step-to-step transitions." In: *Exercise and sport sciences reviews* 33.2, pp. 88–97 (cit. on p. 9).

Kuppuswamy, Naveen and Cristiano Alessandro (2011). "Impact of Body Parameters on Dynamic Movement Primitives for Robot Control". In: *Procedia Computer Science* 7, pp. 166–168 (cit. on p. 18).

Laumanns, Marco et al. (2002). "Combining Convergence and Diversity in Evolutionary Multiobjective Optimization". In: *Evolutionary Computation* 10.3, pp. 263–282 (cit. on p. 51).

Lee, Ki Nam and Young Jae Ryoo (2014). "Walking Pattern Tuning System Based on ZMP for Humanoid Robot". en. In: *International Journal of Humanoid Robotics* 11.04, p. 1442001 (cit. on pp. 21, 29).

Liu, Chengju, Danwei Wang, and Qijun Chen (2013). "Central Pattern Generator Inspired Control for Adaptive Walking of Biped Robots". English. In: *IEEE SMC 2013* 43.5, pp. 1206–1215 (cit. on pp. 2, 24, 28, 29).

Manko, David J. (1992). *A General Model of Legged Locomotion on Natural Terrain*. Boston, MA: Springer US (cit. on p. 47).

Marder, E and D Bucher (2001). "Central pattern generators and the control of rhythmic movements." In: *Current biology : CB* 11.23, R986–96 (cit. on p. 16).

Marder, Eve et al. (2005). "Invertebrate Central Pattern Generation Moves along". In: *Current Biology* 15.17, R685–R699 (cit. on p. 15).

Mass of body segment - calculation. URL: http://biomech.ftvs.cuni.cz/pbpk/kompendium/biomechanika/geometrie_hmotnost_vypocet_en (visited on 03/26/2017) (cit. on p. 148).

Matos, Vítor (2013). "A Bio-Inspired architecture for adaptive quadruped locomotion over irregular terrain". PhD thesis. Universidade do Minho (cit. on pp. 16, 79, 160).

Matos, Vitor and Cristina Santos (2012). "Central Pattern Generators with phase regulation for the control of humanoid locomotion". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, pp. 134–139 (cit. on pp. 81, 160).

Matos, Vitor and Cristina P. Santos (2014). "Towards goal-directed biped locomotion: Combining CPGs and motion primitives". In: *Robotics and Autonomous Systems* 62.12, pp. 1669–1690 (cit. on p. 79).

Matsubara, Takamitsu, Sang-Ho Hyon, and Jun Morimoto (2011). "Learning parametric dynamic movement primitives from multiple demonstrations". In: *Neural Networks* 24.5, pp. 493–500 (cit. on p. 19).

Maurice, Pauline (2015). "Virtual ergonomics for the design of collaborative robots". PhD thesis. Universit{é} Pierre et Marie Curie (cit. on pp. 149, 151).

Maurice, Pauline et al. (2017). "Human-oriented design of collaborative robots". In: *International Journal of Idustrial Ergonomics* 57, pp. 88–102 (cit. on p. 145).

McCown-McClintick, B. E. and G. D. Moskowitz (1998). "The Behavior of a Biped Walking Gait on Irregular Terrain". In: *The International Journal of Robotics Research* 17.1, pp. 43–55 (cit. on p. 2).

Meng, Qinggang and Mark Lee (2009). "Empathy between Human and Home Service Robots". In: *2009 IITA International Conference on Control, Automation and Systems Engineering (case 2009)*. IEEE, pp. 220–224 (cit. on p. 7).

Merlhiot, X. et al. (2012). "The XDE mechanical kernel: Efficient and robust simulation of multibody dynamics with intermittent nonsmooth contacts." In: *Proc. of the 2nd Joint International Conference on Multibody System Dynamics* (cit. on pp. 50, 148).

Micaelli, Alain, Sébastien Barthélemy, and Joseph Salini. *Arboris-python*. URL: https://github.com/salini/arboris-python (cit. on p. 63).

Michalewicz, Zbigniew and David B. Fogel (2004). *How to Solve It: Modern Heuristics*. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 24).

Michel, Olivier (2004). "Webots: Professional Mobile Robot Simulation". In: *International Journal of Advanced Robotic Systems* 1(1), pp. 39–42 (cit. on pp. 50, 153).

Miettinen, Kaisa (1998). *Nonlinear Multiobjective Optimization*. Vol. 12. International Series in Operations Research & Management Science. Boston, MA: Springer US (cit. on p. 35).

Milutinović, Dejan and Jacob Rosen, eds. (2013). *Redundancy in Robot Manipulators and Multi-Robot Systems*. Vol. 57. Lecture Notes in Electrical Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 1).

Mitchel, Melanie (1998). *An Introduction to Genetic Algorithms*. MIT Press (cit. on p. 21).

Mockus, Jonas (1989). *The Bayesian approach to global optimization*. 1st ed. Vol. 5. Springer Netherlands, pp. 297–312 (cit. on p. 22).

Morecki, A. (1997). "Modelling and Simulation of Human and Walking Robot Locomotion". In: *Human and Machine Locomotion*. Vienna: Springer Vienna, pp. 1–78 (cit. on p. 9).

Morimoto, Jun et al. (2006). "Modulation of Simple Sinusoidal Patterns by a Coupled Oscillator Model for Biped Walking." In: *ICRA 2006*, pp. 1579 –1584 (cit. on p. 18).

Morimoto, Jun et al. (2007). "Improving humanoid locomotive performance with learnt approximated dynamics via Gaussian processes for regression". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4234–4240 (cit. on pp. 23, 29).

Morisawa, Mitsuharu et al. (2014). "Biped locomotion control for uneven terrain with narrow support region". In: *2014 IEEE/SICE International Symposium on System Integration*. IEEE, pp. 34–39 (cit. on p. 47).

Mouret, J.-B. and S Doncieux (2010). "SFERESv2: Evolvin' in the Multi-Core World". In: *Proc. of IEEE CEC 2010*, pp. 4079–4086 (cit. on pp. 52, 58, 74, 76).

Mouret, Jean-Baptiste and Jeff Clune (2015). "Illuminating search spaces by mapping elites". In: *http://arxiv.org/abs/1504.04909*. arXiv: 1504.04909 (cit. on pp. 26, 51).

Nwokah, Osita D. I. and Yildirim. Hurmuzlu (2002). *The Mechanical systems design handbook : modeling, measurement, and control*. CRC Press, p. 839 (cit. on p. 10).

ODE Wiki. URL: http://ode-wiki.org/wiki/index.php?title=Main_Page (visited on 03/20/2017) (cit. on p. 50).

Oliveira, Miguel et al. (2013). "Multi-objective parameter CPG optimization for gait generation of a biped robot". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3130–3135 (cit. on p. 81).

Open Design Lab Tools: Proportionality Constant Calculator. URL: http://www.openlab.psu.edu/tools/calculators/proportionalityConstant (visited on 03/26/2017) (cit. on p. 148).

Orlovsky, Grigori, T. G. Deliagina, and Sten Grillner (1999). *Neuronal Control of LocomotionFrom Mollusc to Man*. Oxford University Press (cit. on p. 15).

Panne, M. van de, E. Fiume, and Z.G. Vranesic (1992). "A controller for the dynamic walk of a biped across variable terrain". English. In: *Proc. of IEEE CDC 1992*. IEEE, pp. 2668–2673 (cit. on pp. 20, 24, 28, 29).

Parmiggiani, Alberto et al. (2012). "The Design of the iCub Humanoid Robot". In: *International Journal of Humanoid Robotics* 09.04, p. 1250027 (cit. on pp. 58, 148, 149).

Paul, C. and J.C. Bongard (2001). "The road less travelled: morphology in the optimization of biped robot locomotion". English. In: *IROS 2001 - IEEE/RSJ*. Vol. 1. IEEE, pp. 226–232 (cit. on pp. 22, 29).

Pearson, Karl (1895). "Note on Regression and Inheritance in the Case of Two Parents". In: *Proceeding of the Royal Society of London*, pp. 240–242 (cit. on p. 54).

Pratt, John W. (1959). "Remarks on Zeros and Ties in the Wilcoxon Signed Rank Procedures". In: *Journal of the American Statistical Association* 54.287, p. 655 (cit. on p. 117).

Raghuwanshi and OG Kakde (2004). "Survey on multiobjective evolutionary and real coded genetic algorithms". In: (cit. on p. 37).

Rios, Luis Miguel and Nikolaos V. Sahinidis (2013). "Derivative-free optimization: a review of algorithms and comparison of software implementations". In: *Journal of Global Optimization* 56.3, pp. 1247–1293 (cit. on pp. 32, 34).

RoboSavvy - Robots. URL: https://robosavvy.com/web/ (visited on 03/26/2017) (cit. on p. 152).

Rockafellar, R. T. (1987). "Linear-Quadratic Programming and Optimal Control". In: *SIAM Journal on Control and Optimization* 25.3, pp. 781–814 (cit. on p. 156).

Rodrigues, L. et al. (1996). "Simulation and control of biped locomotion-GA optimization". English. In: *Proc. of IEEE CEC 1996*. IEEE, pp. 390–395 (cit. on pp. 22, 29).

Rossignol, S., Réjean Dubuc, and Jean-Pierre Gossard (2006). "Dynamic Sensorimotor Interactions in Locomotion". In: *Physiological Reviews* 86.1, pp. 89–154 (cit. on pp. 16, 27).

Russell, Stuart J. (Stuart Jonathan), Peter. Norvig, and Ernest. Davis (2010). *Artificial intelligence : a modern approach*. Prentice Hall, p. 1132 (cit. on p. 33).

Salini, J (2012). "Dynamic control for the task/posture coordination of humanoids: toward synthesis of complex activities". These de doctorat. Paris, France: Université Pierre et Marie Curie (cit. on pp. 14, 156).

Salini, Joseph, Vincent Padois, and Philippe Bidaud (2011). "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions". English. In: *IProc. of IEEE ICRA 2011*. IEEE, pp. 1283–1290 (cit. on pp. 13–15, 26, 63, 156).

Sandini, Giulio, Giorgio Metta, and David Vernon (2007). "50 Years of Artificial Intelligence". In: ed. by Max Lungarella et al. Vol. 4850. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. The iCub c (cit. on p. 4).

Schaal, Stefan, Peyman Mohajerian, and Auke Ijspeert (2007). "Dynamics systems vs. optimal control — a unifying view". In: *Progress in brain research*. Vol. 165, pp. 425–445 (cit. on p. 19).

Siegwart, Roland., Illah Reza Nourbakhsh, and Davide. Scaramuzza (2011). *Introduction to autonomous mobile robots*. MIT Press, p. 453 (cit. on p. 8).

Slatton, Andrew et al. (2008). *Integrating a Hierarchy of Simulation Tools for Legged Robot Locomotion*. Tech. rep. University of Pennsylvania, pp. 474–479 (cit. on pp. 21, 27).

Snyman, Jan (2005). *Practical Mathematical Optimization*. Vol. 97. Applied Optimization. New York: Springer-Verlag (cit. on p. 32).

Taga, G., Y. Yamaguchi, and H. Shimizu (1991). "Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment". In: *Biological Cybernetics* 65.3, pp. 147–159 (cit. on p. 17).

Taga, Gentaro (1998). "A model of the neuro-musculo-skeletal system for anticipatory adjustment of human locomotion during obstacle avoidance". In: *Biological Cybernetics* 78.1, pp. 9–17 (cit. on pp. 17, 20, 27).

Talbi, El-Ghazali (2009). *Metaheuristics : from design to implementation*. John Wiley & Sons, p. 593 (cit. on p. 32).

Tomoiagă, Bogdan et al. (2013). "Pareto Optimal Reconfiguration of Power Distribution Systems Using a Genetic Algorithm Based on NSGA-II". In: *Energies* 6.3, pp. 1439–1455 (cit. on p. 36).

Torricelli, Diego et al. (2016). "Human-like compliant locomotion: state of the art of robotic implementations". In: *Bioinspiration & Biomimetics* 11.5, p. 051002 (cit. on p. 15).

Voloshina, Alexandra S et al. (2013). “Biomechanics and energetics of walking on uneven terrain.” In: *The Journal of experimental biology* 216.Pt 21, pp. 3963–70 (cit. on p. 2).

Vukobratovic, M and B Borovac (2004). “Zero-moment point — thirty five years of its life”. In: *International Journal of Humanoind Robots* 1.1, pp. 157–173 (cit. on pp. 9, 10, 12).

Vukobratovic, Miomir et al. (1990). *Biped locomotion: dynamics, stability, control, and application* (cit. on pp. 1, 8–10).

Wang, Jack M. et al. (2010). “Optimizing walking controllers for uncertain inputs and environments”. In: *ACM SIGGRAPH 2010 papers on - SIGGRAPH ’10*. Vol. 29. 4. New York, New York, USA: ACM Press, p. 1 (cit. on pp. 2, 3, 25, 29).

Wang, Jack M et al. (2012). “Optimizing Locomotion Controllers Using Biologically-Based Actuators and Objectives.” In: *ACM transactions on graphics* 31.4 (cit. on pp. 20, 29).

Westervelt, Eric R. et al. (2007). *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, p. 528 (cit. on p. 1).

Wieber, Pierre-Brice (2002). “On the stability of walking systems”. In: (cit. on p. 9).

Wieber, Pierre-brice (2006). “Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations”. English. In: *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 137–142 (cit. on pp. 13–15, 26).

Wilcoxon, Frank (1945). “Individual Comparisons by Ranking Methods”. In: *Biometrics Bulletin* 1.6, p. 80 (cit. on pp. 116, 118).

Wu, Jia-chi and Zoran Popović (2010). “Terrain-adaptive bipedal locomotion control”. In: *ACM SIGGRAPH 2010*. Vol. 29. 4. ACM Press, p. 1 (cit. on pp. 22, 29).

Zheng, Yu and Katsu Yamane (2011). “Optimization and control of cyclic biped locomotion on a rolling ball”. English. In: *Proc. of IEEE-RAS Humanoids 2011*. IEEE-RAS, pp. 752–759 (cit. on pp. 21, 29).

Zhou, Xuefeng et al. (2013). “Stability of biped robotic walking with frictional constraints”. In: *Robotica* 31.04, pp. 573–588 (cit. on p. 49).

Zitzler, Eckart, Kalyanmoy Deb, and Lothar Thiele (2000). “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”. In: *Evolutionary Computation* 8.2, pp. 173–195 (cit. on p. 51).