Learning Safe Controllers for Motion Generation in Redundant Robots

Valerio Modugno

Sapienza - Università di Roma

Doctoral Thesis

Learning Safe Controllers for Motion Generation in Redundant Robots

Author:
Valerio Modugno

Supervisor:
Prof. Giuseppe Oriolo
Dr. Serena Ivaldi

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Diaprtimento di Informatica Automatica e Gestionale - Antonio Ruberti

September 2017

Declaration of Authorship

I, Valerio MODUGNO, declare that this thesis titled, 'Learning Safe Controllers for Motion Generation in Redundant Robots' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"The Terminator' is not an impossibility. I think that symbolizes the downside of artificial intelligence ... but technology has a big downside in general. There is a bigger downside to not pursuing it."

Ray Kurzweil, 2011

SAPIENZA - UNIVERSITÀ DI ROMA

Abstract

Facoltà di Ingegneria
Dipartimento di Informatica Automatica e gestionale-Antonio Ruberti

Doctor of Philosophy

Learning Safe Controllers for Motion Generation in Redundant Robots

by Valerio MODUGNO

One of the key problems in planning and control of redundant robots is the fast generation of controls when multiple tasks and constraints need to be satisfied. In the literature, this problem is classically solved by multi-task prioritized approaches, where the priority of each task is determined by a weight function, or with trajectory optimization techniques. In this thesis we propose a framework that, through the combination of machine learning and control theory approach, can be efficiently applied both for multi task prioritization and trajectory optimization to automatically find optimal behaviour. First we learn the temporal profiles of the task priorities, represented as parametrized weight functions: we automatically determine their parameters through a constrained stochastic optimization procedure. Then we extend the proposed method for trajectory optimization scenario where we learn the task trajectories for whole-body balancing tasks. In both cases we ensure that the optimized movements are safe and never violate any of the robot and problem constraints. For this purpose we compare three constrained variants of CMA-ES on several benchmarks, among which two are new robotics benchmarks of our design using the KUKA LWR. We retain (1+1)-CMA-ES with covariance constrained adaptation [30] as the best candidate to solve our problems. In order to tackle the limitations of the algorithm described in [30], in this thesis we propose an extension of (1+1)-CMA-ES with Constrained Covariance Adaptation (CCA) that addresses all the issue that affects the learning module of our framework. We show the effectiveness of the proposed framework for the task priority learning on a simulated 7 DOF Kuka LWR and both a simulated and a real Kinova Jaco arm. We compare the performance of our approach to a state-of-the-art method based on soft task prioritization, where the task weights are typically hand-tuned. Then we apply our method on two whole-body experiments with the iCub humanoid robot to show its scalability property. Finally we test our learning framework to the prioritized whole-body torque controller of iCub, to optimize the robot's trajectory for standing up from a chair.

Acknowledgements

Firstly i would like to express my sincere gratitude to my supervisor, Prof. Giuseppe Oriolo, for his guidance and support and for the great intellectual and research freedom that he granted me since the beginning of my PhD.

I would like to thank my supervisor, Dr. Serena Ivaldi that gave me a tremendous research opportunity inside the CoDyCo project framework, and with whom I established a fruitful collaboration that stretched across the Europe.

I would like to thank the reviewers of this manuscript, Dr. Paolo Robuffo Giordano and Dr. Jean-Baptiste Mouret for their insightful comments and suggestions.

I would like to thank the Larsen group in Inria Grand-est, Nancy, its director, Dr. François Charpillet, and all the colleagues and friends there. It was one of the best experience of my life both professionally and personally.

I would like to thank the Intelligent Autonomous System team at Darmstadt University and it's head, Prof. Jan Peters, for letting me experience a research environment that was a source of inspiration for my scientific path.

I would like to thank all my colleagues and friends at the DIAG department and all the people (present and past) of the robotics laboratory in Rome that helped me to overcome the toughest moments of my PhD.

I would like to thank my beautiful girlfriend, Lauren, that has been with me since the beginning of this incredible journey demonstrating huge comprehension for the unconventionalities that characterize the life of a PhD student.

Last but not least i would like to thank my family, my father, my mother, my brother and my grandma; I'm grateful and thankful to them because they moulded me into the person that i am today.

Contents

List of Figures

List of Tables

Abbreviations

AV	**A**utonomous **V**ehicle
CoM	**C**enter **o**f **M**ass
DoF	**D**egree **o**f **F**reedom
QP	**Q**uadaratic **P**rogramming
UF	**U**nified **F**ramework
RUF	**R**egularized **U**nified **F**ramework
RBF	**R**adial **B**asis **F**unction
CMA-ES	**C**ovariance **M**atrix **A**daptation **E**volution **S**trategy
CCA	**C**ovariance **C**onstrained **A**daptation
CMA-ES-VC	**C**ovariance **M**atrix **A**daptation **E**volution **S**trategy **V**anilla **C**onstraints
CMA-ES-AC	**C**ovariance **M**atrix **A**daptation **E**volution **S**trategy **A**daptive **C**onstraints
GP	**G**aussian **P**rocess
BO	**B**ayesian **O**ptimization
CBO	**C**onstrained **B**ayesian **O**ptimization
EI	**E**xpected **I**mprovement
ECI	**E**xpected **C**onstrained **I**mprovement
GRF	**G**round **R**eaction **F**orce
CoP	**C**entre **o**f **P**ressure
ZMP	**Z**ero **M**oment **P**oint

Symbols

${}^{a}(\cdot)_{b}$	point b expressed in frame a
I	inertial frame
${}^{I}\mathbf{p}_{B}, {}^{I}R_{B},$	base position and orientation in I frame
$\mathbf{q}_{j}$	joint positions
$\mathbf{M}$	generalized inertia matrix
$\mathbf{C}$	coriolis and centrifugal vector
$\mathbf{G}$	gravity vector
$\mathbf{J}_{C_k}$	$k-th$ contact Jacobian
$\mathbf{J}_{(\cdot)}$	robot Jacobian
$\mathbf{f}$	external wrench
$\mathbf{h}$	centroidal momentum
$\phi(\cdot)$	objective function
n_{IC}	# of inequality constraints $g_i(\cdot)$
n_{EC}	# of equality constraints $h_i(\cdot)$
$n_C = n_{IC} + n_{EC}$	total number of constraints
$\mathbf{\Pi} \subseteq \mathbb{R}^{n_P}$	parameter space of the RBFs network
$\boldsymbol{\pi}_k \in \mathbf{\Pi}$	k-th candidate at the current generation
K	total number of candidates for each generation
K_e	number of best candidates or *elites*
$\boldsymbol{\pi}_{1:K_e}$	best candidates of the current generation
$\mathcal{N}(\bar{\boldsymbol{\pi}}, \mathbf{\Sigma})$	Gaussian distribution with mean $\bar{\boldsymbol{\pi}}$ and covariance $\mathbf{\Sigma}$
σ^2	step size
$\hat{l}(\boldsymbol{\pi}_k)$	penalty factor
$\mathbb{1}_{\{\cdot\}}$	indicator function
$\mathbb{E}(\cdot)$	average value

$\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$	gaussian process
$m(\mathbf{x})$	gaussian process mean
$k(\mathbf{x}, \mathbf{x}')$	gaussian process variance

Dedicated to my family

Chapter 1

Introduction

According to Klaus Schwab, founder and executive chairman of the World Economic Forum [8], today we are on the brink of a new industrial revolution that will radically change our society from the ground up. Our society is going through a great number of changes that has never been experienced in the history of humankind. Today we ignore how this technological explosion will develop but we know for sure that we are going to experience a technological and societal step forward that will flourish at an exponential pace in the way that many futurologists, like Ray Kurzweil, in recent years refer to this as a self sustained process that will push humanity to the technological *singularity* [9]. In this context we do not bolster the most far-fetched ideas of the Singularitarianism movement (such as the creation of superintelligence or the same idea of technological singularity) but, in accordance with this positivistic way of thinking, we uphold the idea of technological fast growth that moves in successive paradigm shifts.

This progression can be seen from the first industrial revolution, where water and steam power were used to mechanize production, to the second, where electric power pushed the creation of mass production. More recently the third industrial revolution, stemmed from electronics and information technology, in less than 20 years brought the world into the fourth revolution. This last revolution is characterized by a technological blending between physical, digital, and biological worlds, and it is strongly dependent on the digitalization process occurred since the middle of the last century.

Velocity, scope, and systems impact are the three main reasons why this technological wave is not just a mere extension of the previous one but is something entirely new that is intended to change our society from the ground up. The speed of current development is unprecedented. The fourth industrial revolution is experiencing a rate of growth that is exponential rather than linear. Moreover, the breadth and depth of technological

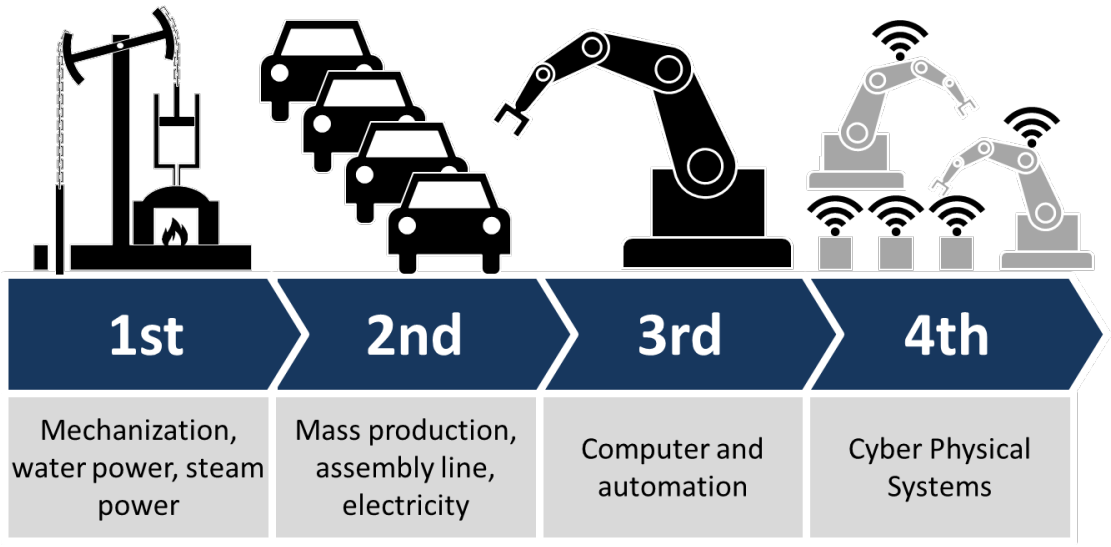


FIGURE 1.1: This illustration shows the sequences of the four industrial revolutions by highlighting their main features. For the fourth revolution by Cyber Physical System we refer to a mechanism that is controlled by computer algorithms and is characterized by a tight integration among them.

innovation is changing every single aspect of our society from production to governance and it is disrupting almost every industry in the entire world.

Today mainstream media grants more and more air time to stories about new arising technologies or innovative treatments for deadly diseases. Nowadays, many technology-related buzzwords reach the general public more than ever. One of these, is *Industry 4.0* a framework that tries to decline the ideas of the fourth industrial revolution in the manufacturing environment.

The entire Industry 4.0 concept is built around the concept of "smart factory". The smart factory is conceived as an independent unit where autonomous systems monitor the physical process inside the factory and provide a full control of the plant. A Smart factory requires high interoperability among devices, sensors and people, and capability to recollect and analyze all the sensory data an provide an autonomous response depending on the current state of the factory, or support humans in making decisions. A fully autonomous productive unit represents the last step on the road towards the industry 4.0 implementation. A "light-out factory" is a place where no human intervention is required in order to function, thus the factory can operate with lights switched off. A fully automatic factory is a place where the raw materials enter as input and finished products leave. Another great example of the effect of the fourth industrial revolution is represented by the rising of the autonomous cars technology. An autonomous car is a machine capable of taking producing real-time decisions based on the current state of the environment. Nowadays this technology has not yet reached the streets of our cities but it has the potential to disrupt the logistics industry and change the life of millions

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

FIGURE 1.2: In this picture are presented the 6 levels of autonomy for AVs, according to the Society of Automotive Engineers (SAE). It is worth noticing the difference between level 2, where the driving task is performed by humans, and level 3 where the automated driving system is responsible for the control of the vehicle. This table is taken from https://www.sae.org/misc/pdfs/automated_driving.pdf

of people. In 2014 an autonomous cars classification with 6 different levels of autonomy was published by SAE international, a US based automotive standardization authority. These levels give a technical and descriptive characterization of each state of autonomy rather than defining a normative framework. Each level introduce a lower bound for the system capabilities. The table is shown in figure 1.2

All these examples point towards the same directions: here the common goal is to implement a level of autonomy that aims to make the human intervention superfluous. Today this goal appears to be closer than ever in the history of the human kind due to the great progress that the scientific community from both public and private institutions is pursuing in two very important sectors of the fourth industrial revolution: Machine Learning and Robotics Control. Just looking at applications for robotics, worldwide, the private sector expenditure for 2017 will total 97.21 billion dollars with a predicted acceleration in spending over the next five years, that will hit 230.7 billion in 2021. The International Data Corporation (IDC) that redacted the report explains that the sustained growth of investments is due to the convergence of robotics and machine learning, that is opening the way to the next generation of intelligent robots for industrial, commercial, and consumer applications. Even if we have witnessed a contraction in the

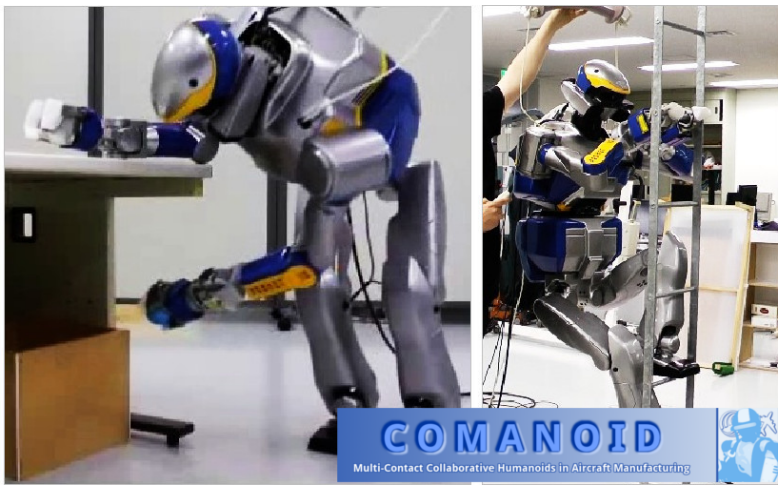


FIGURE 1.3: In this figure an HRP-4 humanoid robot is involved in two different working scenarios that can occur inside a plane factory.

public funding for research worldwide after the financial crisis in 2008, the public institutions are still playing a prominent role in the support of robotics R&D. In particular, in the last decade the European Union, through the FP7 and horizon 2020 research programs has already funded more than forty thousand projects with a combined total investment of 130 billion euros over 15 years (FP7 with 50 billions from 2008 to 2013 and H-2020 with 80 billions (estimated) from 2014 to 2020) with more than 1472 of them related to robotics (according to the Community Research and Development Information Service - CORDIS).

Among this multitude of projects, two of them, COMANOID and CoDyCo, highlight many interesting aspects about the current state of research in the robotics field. Citing from the main web page of the project: "*COMANOID aims at deploying humanoid robots to achieve tasks that have been identified by Airbus Group in aircraft assembly operations. The project focuses on showing precise accessibility (namely into areas where wheeled robots cannot be deployed) through whole body multi-contact planning motion with advanced embedded 3D dense SLAM localization and visuo-force servoing capabilities. Because the robots evolve in human worker co-localized spaces, safety issues will be specifically accounted for*". In CoDyCo, the proposers of the project aim to advance the current state of whole body motion control for humanoids and provide control and cognitive robust goal directed tasks with multiple contacts interaction point with the environment. In this project there is an explicit reference to the combination of machine learning and control theory in order to advance the corpus of methods for whole body control. Both the projects are concerned with the development of controllers that allow the humanoid robot to interact with a dynamical environment where multiple exchanges of forces through contacts are considered to solve the tasks. In COMANOID there is an explicit reference to the satisfaction of safety conditions during the execution of the

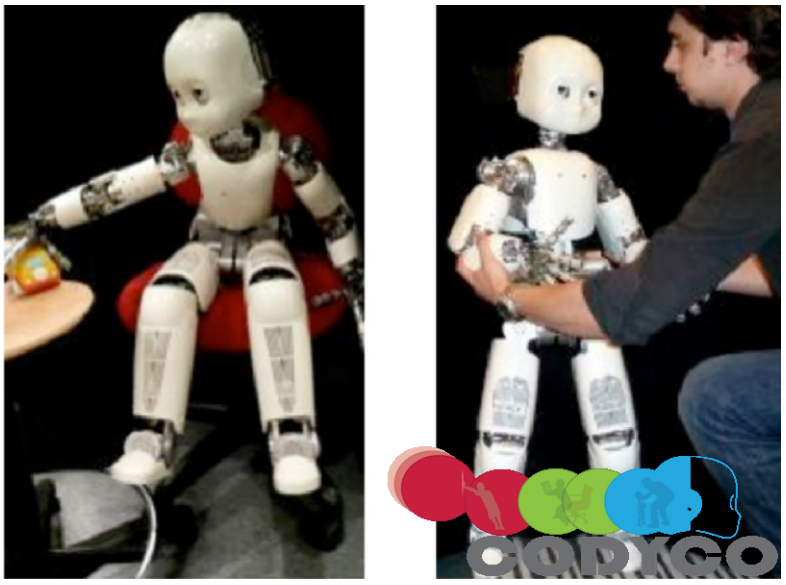


FIGURE 1.4: In this picture two of the final demos that where conceived for the CoDyCo European Project. This scenario involves multiple contacts with the environment and a force interaction with a human operator.

task, while in CoDyCo a smart combination of control and machine learning is considered one of the key factors to advance the whole body control strategies. The research proposed in this manuscript is grounded in this theoretical framework: our work wants to leverage from both machine learning and control theory to define new strategies in order to achieve better performances for humanoid whole body control applications and to provide solutions for complex tasks involving contacts with the environments. We want to do all of that by satisfying all the safety requirements that are necessary in order to avoid dangerous outcomes both for the humanoids and the humans involved in the interactions. In the next sections we will describe the problem of safe motion generation and in the last section we will define the contribution of this thesis.

1.1 Safe Motion Generation

The generation of complex behaviour for highly redundant robots is a challenging problem which solution is a necessary step towards the deployment of fully autonomous robotic platforms in every economic sector. Exploiting the redundancy in robotic systems to simultaneously fulfil a set of tasks is a classical problem for complex manipulators and humanoid robots [10, 11].

Depending upon the global task that is requested to solve, several controllers have been proposed in the literature. Multi-task coordination and trajectory optimization are the classical ways to face the redundancy resolution problem.

For prioritized multi-task controllers there are two main approaches. The first is based on *strict task priorities*, where a hierarchical ordering of the tasks is defined: critical tasks (or tasks that are considered as more important) are fulfilled with higher priorities, and low-priority tasks are solved in the null-space of the higher priority tasks [12, 13]. Another example is represented by Sentis and Khatib [14] where the authors defined three levels of hard priorities *i.e.*, constraints of utter importance (such as contacts, near-body objects, joint-limits and self-collisions), operational tasks demands (*i.e.*, manipulation and locomotion) and adaptable postures (*i.e.*, the residual motion). However, in many contexts, it is difficult to organize the tasks in a stack and pre-define their relative importance in form of priorities. When priorities are strict, a higher-priority task can completely block lower-priority tasks, which can result in movements that are not satisfactory for the robot mission (*i.e.*, its "global" task). Another issue concerns the occurrence of discontinuities in the control law due to sudden changes in the prioritization [15].

The second method is based on *soft task priorities*, where the solution is typically given by a combination of weighted tasks [16]. The importance or "soft priority" of each individual task is represented by a scalar weight function, which evolves in time depending on the sequencing of the robot actions. By tuning the time-dependent vector of scalar weights, the global robot motion can be optimized. In simulation studies, it was shown that adapting these weights may result in a seamless transition between tasks (*i.e.*, reaching for an object, staying close to a resting posture and avoiding an obstacle), as well as in continuous task sequencing [17]. However, the simultaneous execution of different elementary tasks with variable soft priorities can lead to incompatibilities that might generate undesired movements or prevent the execution of some tasks. These issues are well explained in [18], where the authors modulate the task weights based on the movement variance to handle incompatibilities during online execution. Moreover, when the number of tasks increases, (this is true especially for whole-body control of humanoid robots), and some tasks related to safety (*e.g.*, balance) are given high priority, it is generally difficult to define suitable task activations. In this case the priorities and their transitions are manually tuned by expert users [17] or defined before-hand [20]. In Liu et al. [17] the authors proposed a generalized projector (GHC) that handles strict and non-strict priorities with smooth transitions when tasks priorities are swapped. Despite the elegant framework, their controller needs again a lot of manual tuning. The evolution of the tasks priorities in time, the timing and the tasks transitions need to be designed by hand. While this approach could still be easy for few tasks and simple robotic arms, it quickly becomes infeasible for complex robots such as humanoids performing whole-body movements that usually require a dozen of tasks and

constraints (*e.g.*, control balance, posture, end-effectors, stabilize head gaze, prevent slipping, control the interaction forces *etc.*).

In our first application of the framework presented in this thesis we propose to automatically learn the task weight functions, described as parametrized functional approximators. The concept of learning the soft priorities can be applied to existing multi-task frameworks, such as the GHC [17]. However, we introduce here a simpler controller based on a regularized version of the Unified Framework for Robot Control (UF) [21] proposed by Peters *et al.*

When the global movement to realize is complex, it is not straightforward to define an a priori granulation of the main task. For example, let us consider a humanoid that must stand up from a chair. This motion, trivial for a human, is very challenging for a humanoid. During the execution of the stand up we want to be sure that the robot produces the right acceleration of its Center of Mass (CoM) while balancing. This necessitates optimizing its posture at each time step. In this example, a significant amount of manual tuning is required to optimize the motion and adapt it to the robot: one can optimize the task priorities/weights, their evolution in time (e.g., their relative priority, or scalar value, or the task transitions), and/or the task trajectories, by acting on the desired trajectories (e.g., desired CoM) or some via-points (e.g., to avoid collisions or potentially unstable configurations). For this application, this kind of motion is characterized by switching contacts (from the legs on the chair to the feet on the ground), physical interaction with the environment and several balance constraints to be verified. Executing this kind of motion with the previous approach (learning task priorities) is hardly possible, because it is not suited to segment a complex movement into phases characterized by different contacts configurations. Therefore we designed a different approach: we consider a controller with a fixed strategy and task priorities, and optimize the task trajectories, i.e. the reference trajectories or desired trajectories of the tasks. In particular for the stand up case, we do not use a weighted QP controller but the prioritized whole-body controller proposed by Nava *et al.* [22], which is capable of performing highly dynamic tasks in iCub [23]. Again for the optimization of the parametrized task trajectories, we employed a black-box stochastic optimization.

Both the application cases proposed so far rely on black box optimization technique. In this thesis we propose a general approach that tries to solve several limitations that afflicts transversally almost all the algorithms proposed in the literature. Many of them display serious structure limitations. They rely on optimal solvers that limit the way that a problem can be formulated. They relies on quadratic cost function and need an analytical description of the environment and the constraints that define the global task that has to be achieved. In order to generalize both the structure of the objective

function and the way to describe the global behaviour that we want to obtain, we decided to employ a black-box optimization method, CMA-ES [24]. Unfortunately this choice comes at a cost: we need to provide a simulation of the global task that we want to solve. It means that the solution that we find has to be optimized offline many times with the aid of a simulation engine.

In this thesis we take a major step in the direction of taking into account constraints satisfaction directly inside the machine learning procedure. Ensuring that the optimization process yields a "safe" solution — where safety means not violating any constraints — becomes mandatory if we want to successfully apply these solutions to a real robot [25]. Even the most recent approaches that are similar to our framework and are based on an iterative policy learning technique that needs many repetitions (rollouts) of the same experiment to find a viable solution, [18, 26–28], poorly address the problem of constraints satisfaction when optimizing the task priorities. For example, in [27], torques are saturated for safety, and joint and velocity limits are introduced as tasks. However, satisfaction of constraints formulated as tasks cannot be ensured, especially in the case of soft tasks prioritization. In [28] the balance constraint is added as an objective to the fitness function, but this is a relaxation of the constraint that does not ensure its satisfaction either. In [26] we used the Covariance Matrix Adaptation-Evolutionary Strategy (CMA-ES) [24], a derivative-free stochastic optimization method that solves non-linear, non-differentiable optimization problems, with death penalties to enforce constraint satisfaction on the solutions. This choice was not efficient in terms of searching for the optimum solution, since the exploration could easily get stuck in a constrained region where the fitness landscape was turned into a plateau. Furthermore, many solutions had to be dropped because of constraints violation. To approach the safety issue, in this thesis we investigate *constrained* stochastic optimization algorithms, and we focus on three variants of CMA-ES: one with *vanilla* constraints, one with adaptive constraints [29] and the (1+1)-CMA-ES with covariance constrained adaptation [30]. We compare these methods with a baseline constrained optimization algorithm, (the *fmincon* function in Matlab). To compare the algorithms, we explicitly look for methods that can find good solutions while ensuring zero constraint violations within a reasonable computation time.

There exist standard benchmarks for constrained optimization, consisting in analytic problems with several variables and constraints and known optimal solutions. For example Arnold & Hansen [30] tested (1+1)-CMA-ES on seven different problems with a number of variables ranging from 2 to 10, and a number of constraints between 1 to 9. However, in robotics the number of constraints usually grows with the number of Degrees of Freedom (DoF) of the robot: for example, with a 7-DoF robot, the joint position range (7×2) and the torque limits (7×2) already introduce 28 constraints.

In humanoids and highly articulated systems, the number of DoF is higher (*e.g.*, 32 DoF for the iCub) and so is the number of constraints. Furthermore, the number of tasks increases with the complexity of the action, especially for bimanual or whole-body movements. It is therefore necessary to design new benchmarks tailored for robotics applications to make a pondered decision about the algorithm that is most suited to solve our problem while ensuring that the constraints are never violated. So we compared the performance of three constrained variants of CMA-ES with *fmincon* on analytic and robotic benchmarks where the latter (RB1,RB2) were been designed designed *ad hoc* to provide a meaningful benchmark for robotics application.

Finally in this thesis we present some preliminary results on a new approach that has been conceived in order to provide a possible solution for the slow convergence rate issue and the feasible starting point issue that affects (1+1)CMA-ES with Constrained Covariance Adaptation (CCA).

1.2 Contribution of this Thesis

In this thesis we gathered together the results of three different works that we published or we submitted for publication. In our first work [26], we tackled a daunting problem for the design of multi task controller. In this context the definition of hierarchies plays a central role to successfully achieve a solution for the objective global task.In literature, this kind of problem is solved using two different approaches: strict priorities and soft priorities. In both cases we need the intervention of an expert to manually design the correct prioritization (both the sequence of tasks and the switching time) to find the correct solution. In this work we proposed an automatic technique to find an optimal solution for the prioritization problem. A controller was presented, based on parametrized soft task priorities and the parameters were learned using CMA-ES a black box Stochastic Optimization technique. We showed the effectiveness of our approach on both a simulated and a real 6 degrees of freedom Kinova Jaco arm, on a goal reaching problem with several elementary tasks. Furthermore, we compare the performance of our controller with the state-of-the-art method GHC proposed by Liu *et al.* [17] on a simulated 7 DoF Kuka LWR arm. Moreover, constraints satisfaction is another issue that affects the problem of multi-task combination. It depends on the fact that many solvers for quadratic programming problems rely on a relaxation procedure that allows for constraints violation. In our second work [31] we proposed to address this issue by extending the previous technique to deal with constraints. In this second work we searched for a constrained black-box technique that provided a solution that satisfies

the constraints without violation. We benchmarked three different extension of CMA-ES [24] that incorporate the capacity to deal with constraints. We found out that the method from [7] satisfied all of our requirements with performance comparable with the other two methods. In our last work [32], we extended the framework by showing that the idea of optimizing some task parameters guaranteeing the constraints satisfaction can be applied not only to learning task priorities but also task trajectories. In this work we combine the learning capabilities of our method with a well established balance controller already applied with success on the iCub Platform [23]. In this work we aim to extend the results on balance [23], leveraging on our framework, to find optimal trajectories on the iCub platform for extremely challenging task like standing up from a chair. In this work we provided a solution in simulation to show the capabilities of the proposed solution. Finally in this thesis we present early results for a (1+1)CMA-ES + CCA extension that aims to solve some of the issues of the original algorithm. In this work we propose a different approach that tries to accelerate the converging rate of (1+1)CMA-ES + CCA leveraging on the information that we collect and store inside a Gaussian Process. Then this information is used inside a Bayesian optimization framework that provides a solution for the feasible starting point that affects (1+1)CMA-ES + CCA in its basic implementation. Here we show some preliminary results where the the proposed algorithm is benchmarked on a set of analytical constrained objective functions and compared with other state of the art methods.

The research in this thesis has been published/submitted for publication in the following papers:

- Modugno, V.; Chervet, U.; Oriolo, G.; Ivaldi, S. (2016) Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization. Proc. IEEE/RAS International Conf. on Humanoid Robots (HUMANOIDS).
- Modugno, V.; Neumann, G.; Rueckert, E.; Oriolo, G.; Peters, J.; Ivaldi, S. (2016) Learning soft task priorities for control of redundant robots. Proc. IEEE International Conf. on Robotics and Automation (ICRA).
- Marichal*, S.; Malaise*, A.; Modugno, V.; Dermy, O.; Charpillet, F.; Ivaldi, S. (2016) One-shot Evaluation of the Control Interface of a Robotic Arm by Non-Experts. Proc. International Conf. on Social Robotics.
- Modugno, V.; Nava, G.; Pucci, D.; Nori, F.; Oriolo, G.; Ivaldi , S. (2017) Safe trajectory optimization for whole-body motion of humanoids. Submitted to IEEE/RAS International Conf. on Humanoid Robots (HUMANOIDS).

Chapter 2

A Brief Literature Survey on Motion Generation

In this chapter we present the most prominent work about motion generation for redundant robots. For the robotics community, motion generation is one of the central topics that aims to provide greater autonomy capabilities for robotic platform. In this compendium we focus our attention on two specific approaches for motion generation:

- Combination of multiple tasks
- Trajectory optimization

2.1 Motion Generation through the Combination of Multiple Tasks

Since the very early days of the robotics research [33], fulfilling multiple operational tasks to achieve a complex behaviour while satisfying constraints is one of the challenges of whole-body control for redundant robots. In the early work on manipulators, the redundancy was resolved by exploiting the geometrical properties of the jacobian. A pseudo inversion of Jacobian allows task prioritization that protects from tasks inconsistencies. Solutions were proposed for velocity [10, 11, 34, 35], acceleration [36–39] and torque [21, 40] controllers. For example, let us consider the humanoid iCub that must fulfil a "global task". The global task can be decomposed as a combination of simpler elementary tasks (for example: control the end-effector, control the pose of a particular link, *etc.*) and constraints that guarantee a condition of feasibility over the generated motions (for example: torque and joint limits, collisions, external forces *etc.*).

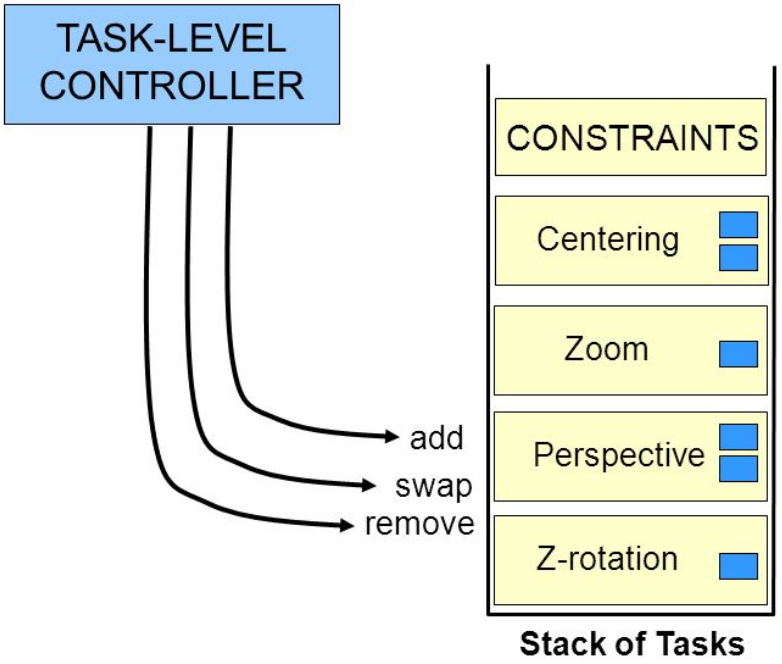


FIGURE 2.1: In this picture is shown the general scheme of the Stack of Tak approaches described in Mansard et al. [1]. On the right the stack of task is presented while on the left the blue box represents the high level controller that can change the stack of task order through three different actions: remove a task, swap two task or add a new task.

More generally, elementary tasks can include tracking desired trajectories, regulating contact forces, controlling the center of mass for balancing, *etc.* Constraints range from mechanical limitations (*e.g.*, joint and torque limits) to safety specifications (*e.g.*, collision avoidance, limiting the exchange of mechanical forces with the environment) and balance keeping for floating base platforms. In literature there are two main approaches for prioritized task controllers. The first is based on *strict task hierarchies*, where a hierarchical ordering of the tasks is defined: critical tasks (or tasks that are considered as more important) are fulfilled with higher priorities, and the low-priority tasks are solved in the null-space of the higher priority tasks [12, 13]. One of the early work for task prioritization is from Siciliano and Slotine [11]. In this work the authors propose a general framework that, given a predefined hierarchy of tasks for an highly redundant structure, provides joint velocity and acceleration references that realize the desired stack of task. The strict prioritization is assured through the projection of each task in the null space of the all the other with higher priority. De Luca and Oriolo [41] proposed a method for a dynamic resolution of redundancy through local optimization, where an optimization based joint decomposition is performed and an efficient computational scheme for real time control is proposed. To avoid the saturation of the joint velocities, the authors present several dynamic criteria that produce acceptable torque inputs.

More recently, Mansard and Chaumette [1] introduce a novel multi-task controller that

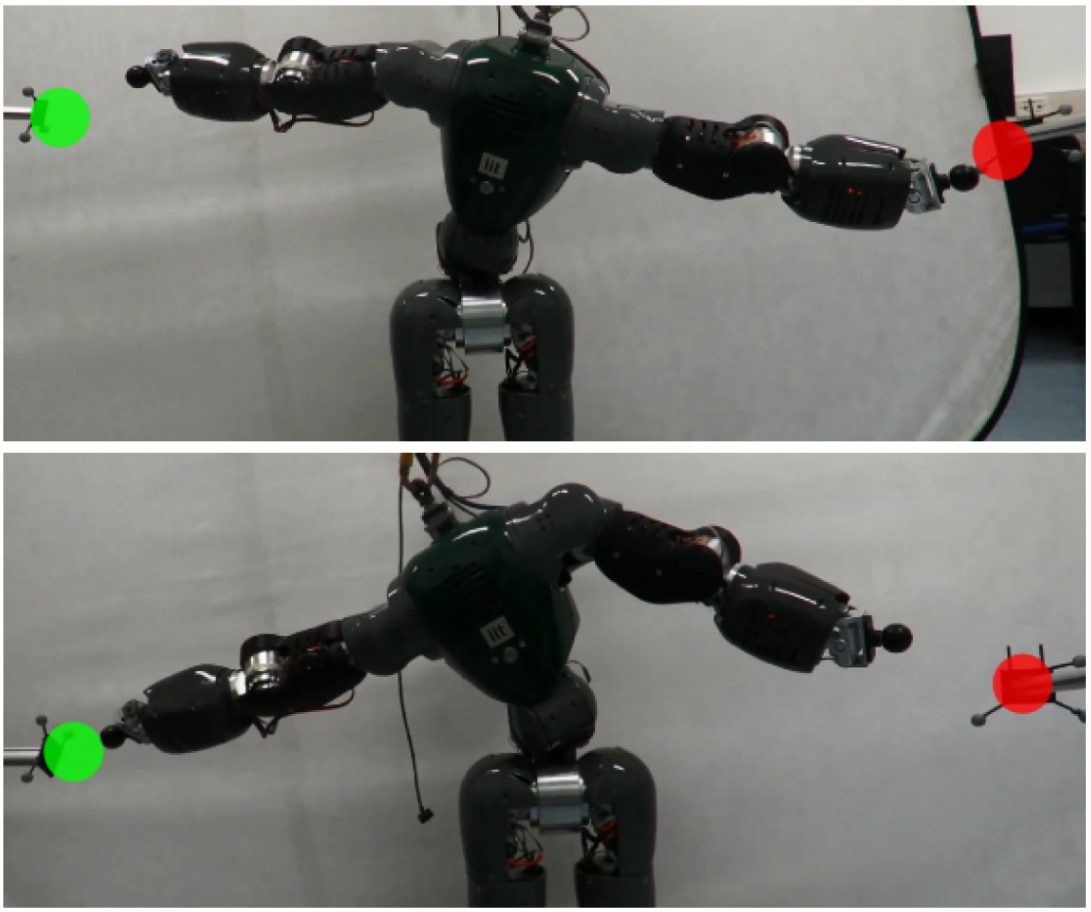


FIGURE 2.2: This picture shows a bimanual movements with two different tasks on a real COMAN humanoid robot. IN this experiment in the first row is shown how the task hierarchies learned through the method described in [2] prioritizes the red marker. IN the second row is shown a reproduction of the task priority model learned by giving more weight to the green marker.

tries to mitigate the fixed hierarchies issue. In their framework each task can be activated or deactivated. When the motion is far away from any constraints the robot can execute the full motion. Otherwise, when the robot is getting closer to a forbidden configuration, an high level task manager intervenes to deactivate one or more tasks. Once the robot overcomes the configuration to avoid all the tasks are restored, although, in many practical contexts, it is difficult to organize the tasks in a stack and define their relative importance in forms of priorities. When priorities are strict, a higher task can completely block lower tasks, which can result in movements that are not satisfactory for the robot mission (e.g., its "global" task). Another issue is the occurrence of discontinuities in the control law due to suddenly changes in the prioritization [15].

The second is based on *soft task hierarchies*, where the solution is typically given by a combination of weighted tasks. With this approach the robot global behaviour will depend from the vector of scalar weights and their evolution in time, [16]. In [17], it

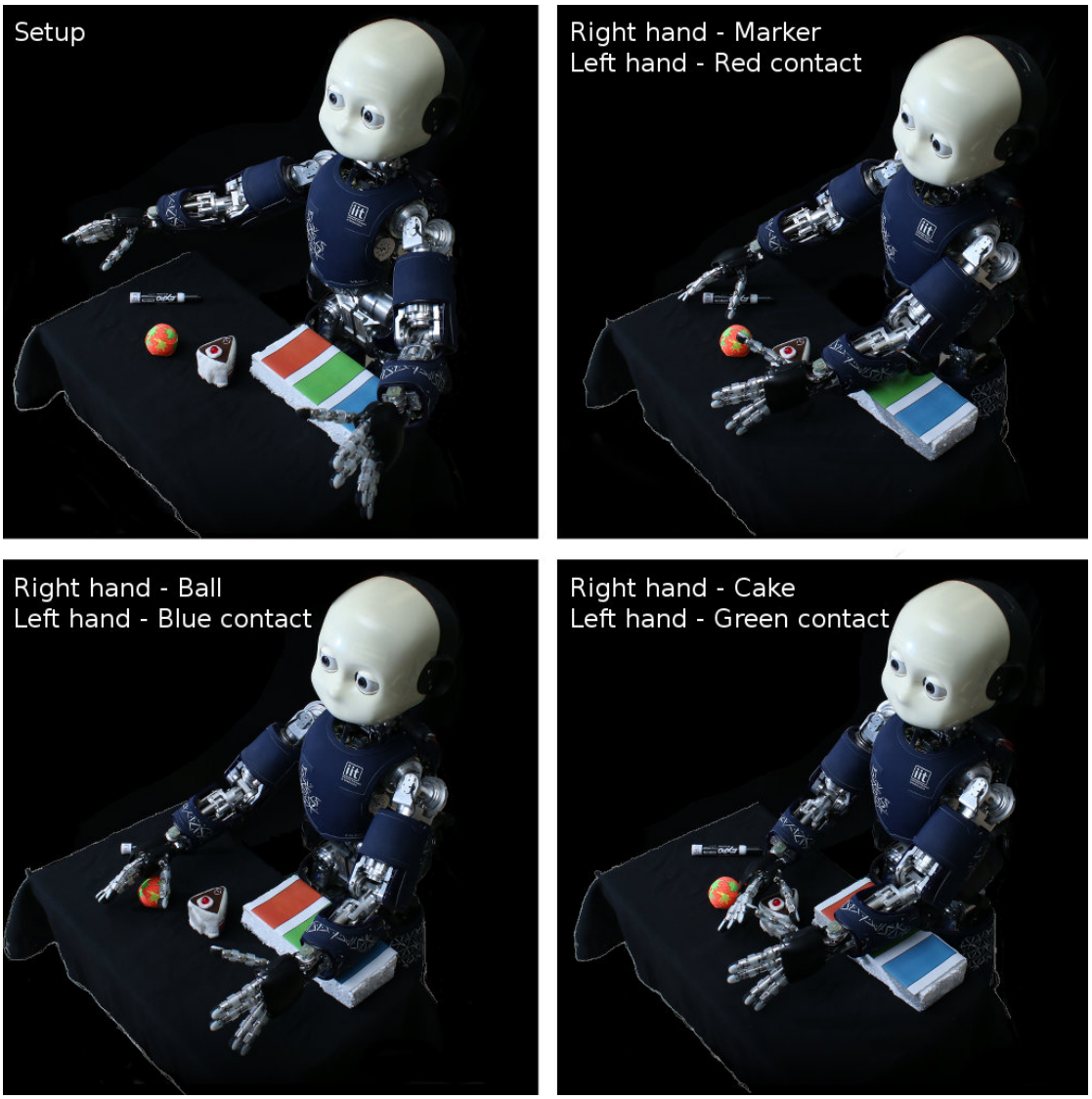


FIGURE 2.3: In this picture is shown an iCub performing a bimanual task (from [3]). In this experiment the icub has to reach an object on the table while supporting himself with the other arm. The iCub has three different supporting areas for the left arm identified by different color. In clockwise order from the top-right three different task combination are presented that are learned through the method in [3].

was shown that adapting these weights may result in a seamless transition between tasks (i.e., reaching for an object, staying close to a resting posture and avoiding an obstacle) and in continuous task sequencing Especially for humanoid robots, that need to perform complex manipulations while satisfying many constraints (e.g., keeping a posture, avoiding obstacles, controlling the contact forces with the environment), soft task hierarchies methods show many interesting features. In these scenarios, the strict control approaches typically require the pre-specification of the task hierarchy.

Recently Silverio et al. [2] proposed a framework based on learning by demonstration for bimanual robots. In this work they employed an extension of task parametrized Gaussian

Mixture Model to learn operational and configuration space constraints and the task priorities as well, given a set of candidate task hierarchies. In order to provide a different task prioritization, we need to design different demonstration for the robot although, in some application scenario, is not easy to design all the possible demonstrations for competing tasks together with all the possible candidate tasks hierarchies.

Parachos et al. [3] presented a probabilistic prioritization framework for tasks represented as Probabilistic Movements Primitives (ProMPs)[42]. The ProMPs are controllers that execute a trajectory defined as a distribution of repeated movements over the same full motion sequence. In this paper the authors extend this framework by adding another feedback for the tasks prioritization and to amend inaccuracies. For the prioritization they exploited the task variance information in time to modulate the priority of each task. This prioritization method can be interpreted as an extension of [18]. Moreover, the authors propose a procedure to learn the trajectory prioritization from demonstration and they showed that this soft prioritization scheme can combine tasks to generate even unseen movements.

2.2 Motion Generation through Trajectory Optimization

Trajectory optimization has been a critical topic in robotics for decades, especially to generate highly dynamic motions. In some cases, the problem of trajectory optimisation has been decoupled from the controller used to execute the trajectory on the robot. For example, in [4] Mordatch *et al.* propose the Contact Invariant Optimization method to synthesize highly complex behaviours, decomposed in different phases with their own set of contacts and weight functions. The problem of whole-body control for redundant robots such as humanoids involves fulfilling multiple operational, posture and force tasks, while satisfying several constraints that guarantee the physical feasibility of the generated motions [43, 44]. The problem is classically solved by prioritized multi-task controllers with strict task priorities [45, 46] or soft task priorities (also called weights) [47, 48]. It is formulated as a QP problem subject to constraints, that is solved at each time step of the control loop [49, 50]. In some cases, the problem of trajectory optimisation has been decoupled from the controller used to execute the trajectory on the robot. To optimize the global robot movement, one classic approach is to fix the desired task trajectories (e.g., they are known in advance, or demonstrated by a human) and optimize the task priorities/weights (including their value in time, their transitions, *etc.*). Very frequently, this optimization is done manually. In [26, 31] we parametrized the task priorities and optimized their parameters to obtain behaviours to maximize a

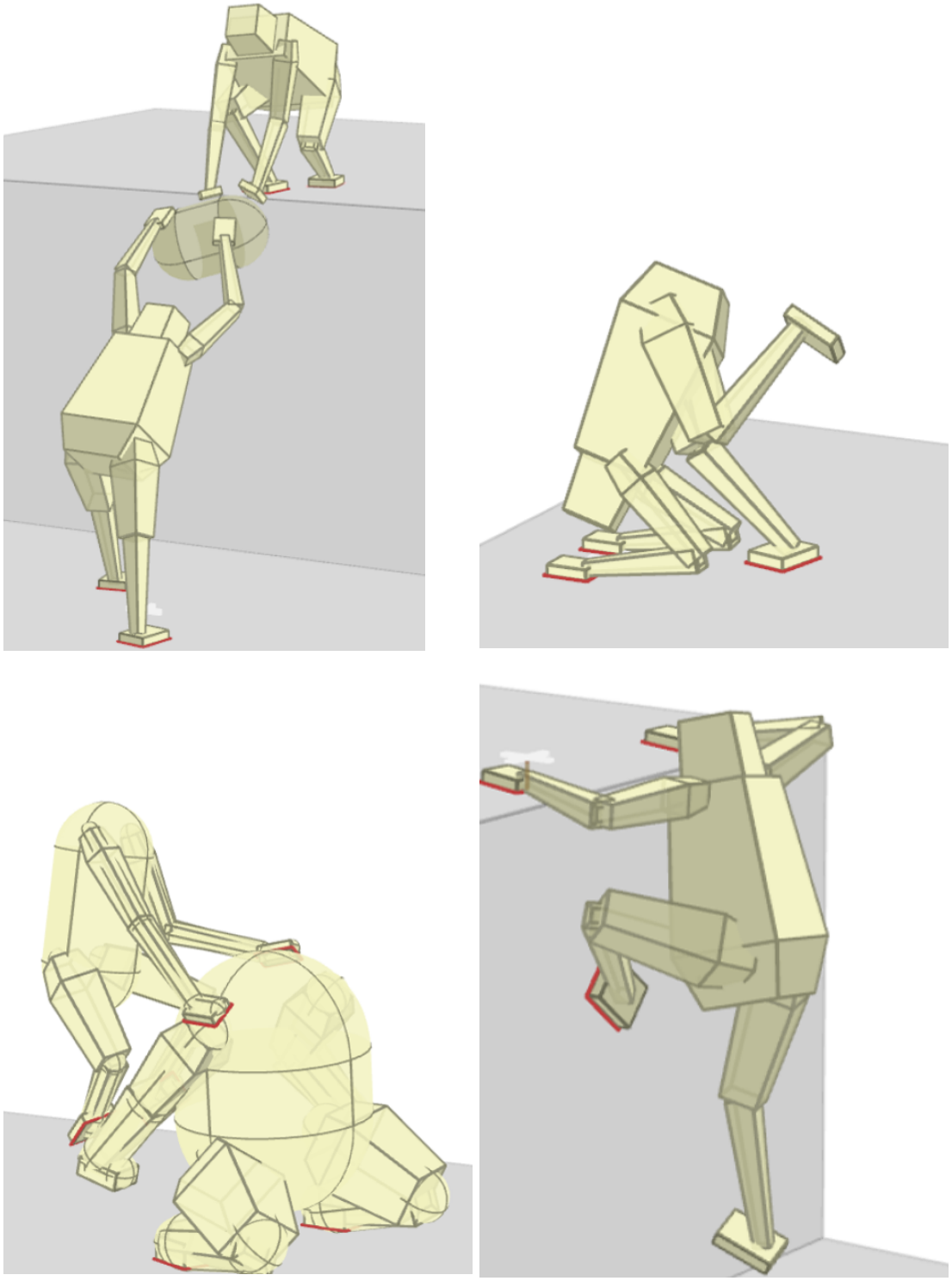


FIGURE 2.4: In this pcitures we show 4 different movements generated with the CIO framework [4]. This technique can deal with a great number of contacts in very different scenario even with robots with different geometric model.

fitness function while satisfying the problem constraints. The Contact Invariant Optimization method [4] belongs to the second group. In this paper the author propose a contact based trajectory optimization approach to synthesize highly complex behaviour for general robot morphology. In this method a complex behaviour is decomposed in different phase and each of them is associated with a set of contacts through the associated set weight functions. In the broad context of trajectory optimization for planning, another powerful framework is the Incremental Optimization for Real-Time Replanning (ITOMP) [51]. The aforementioned paper presents an optimal planning framework that can avoid moving obstacle by interleaving planning and execution. Both in [4] and [51], the constraints of the problem are managed as an additional cost inside the objective function. This might lead to solutions that are not feasible with catastrophic outcomes if such occurrences are not properly managed. Alternatively the trajectory optimization is reframed as an optimal control problem. For locally linearized system with quadratic cost function and gaussian noise we have [52], [53], [5]. In the work of Todorov et al [52], a new method is presented that iteratively improves an optimal controller that locally optimizes a linearisation of a general non linear system around the current trajectory. Subsequently this method has been extended in [53] where the ILQG from [52] is used as optimization routine inside a Model Predictive Control scheme. Finally in [5] the author propose a generalized version of Differential Dynamic Programming a method that provides a second order approximation of the Bellman equation (where ILQG employs a first order approximation of the same expression). These trajectory optimization methods provide explicit means to manage constraints on the control inputs with backtracking line search for [52], [53], or by solving quadratic program with box constraints for each time step in [5]. None of them combat the state space constraint issue in an explicit way, so there is no guarantee of constraints satisfaction in scenarios where real robot are involved. A broader extension of ILQG is presented in Toussaint [54]. In this work the author recasts the problem by conditioning a probabilistic trajectory on desired criteria, proving that the maximum likelihood of the conditioned probability correspond to the optimal solution of the ILQG case. Finally an extension to non gaussian noise model is introduced through the application of approximate inference techniques. Even if [54] results in a much larger set of problem that can be solved it reduces the protections offered against constraints with respect to ILQG. Another recent work that claims to extends the results from [52] is [55]. In this work the linear approximation of the model dynamic is dropped and replaced with a model free approach based on a quadratic Q-function. The idea behind the proposed algorithm is that, at each iteration, an estimation of the Q-fuction and the state distribution are provided by evaluating the current policy several times. After that, using these quantities, a new policy is computed as a solution of a constrained optimization problem that maximize the average of the current Q-function by satisfying a KL upper bound (that act as step size) and a entropy lower bound on

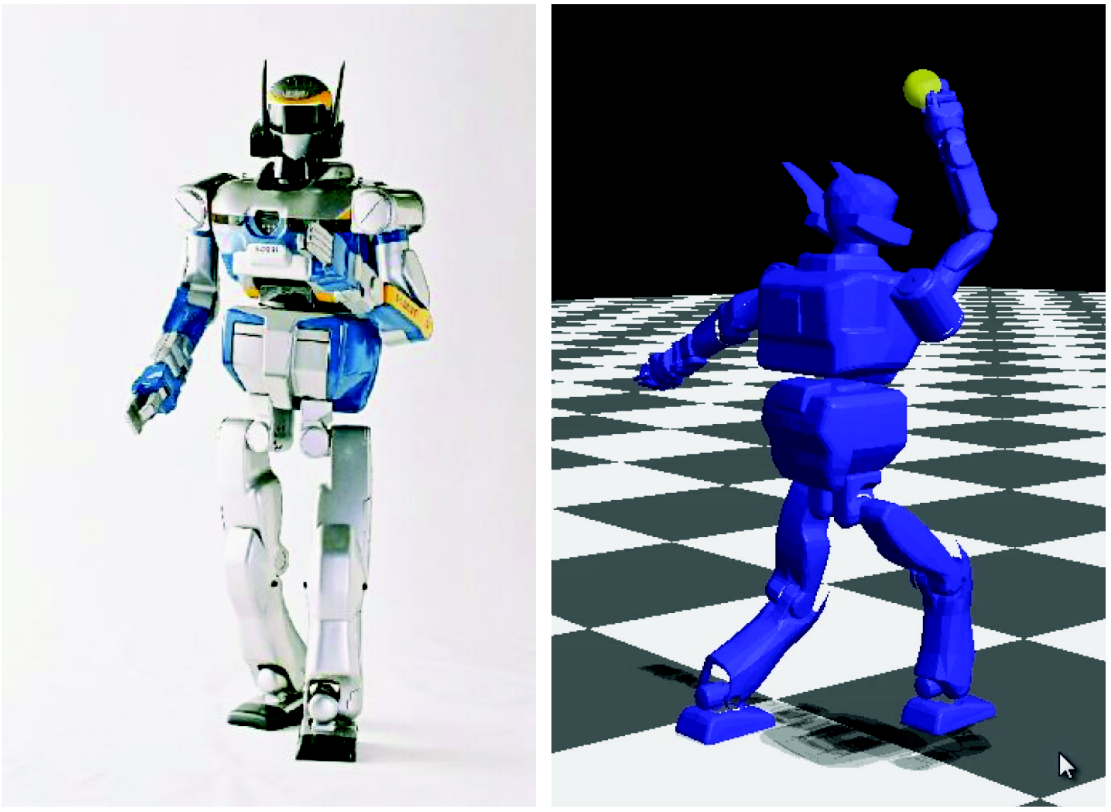


FIGURE 2.5: In this picture is shown a simulated results from the application of the method proposed in [5]. In particular on the right is shown a robot that is keeping the balance while performing a reaching task with the right arm. The picture on the left shows a humanoid robot called HRP-2.

the new policy. The proposed method as the others presented in this section ignores the existence of constraints defined in the state and action spaces. In [6] the authors are fully aware of the constraints issue. They propose a framework that combines optimal control and trajectory planning to compute a set of policies that stabilizes, from any starting position, a non-linear system toward a goal state. The motion planning module generates an open-loop trajectory for the system that satisfies the control-state constraints. Then the feedback module assures local stability around the nodes of the computed trajectory. The union of the controllable region around each ball produces the so called *funnel*. All the states in the funnel under the optimal control policy reach the goal and satisfy all the problem's constraints. The main limitations regard the problem's structure where only quadratic cost functions are allowed and the objective difficulties to extend the framework for physical interaction with the environment. The framework that we propose in this thesis resolve all the drawbacks that affect the other methods. Our learning module relies on a derivative free constrained optimization method that poses no restriction on the structure of the problem and guarantees constraints satisfaction both in the state and in the control space. The integration of the learning module with established torque controllers lets us obtain solutions that are robust due to the

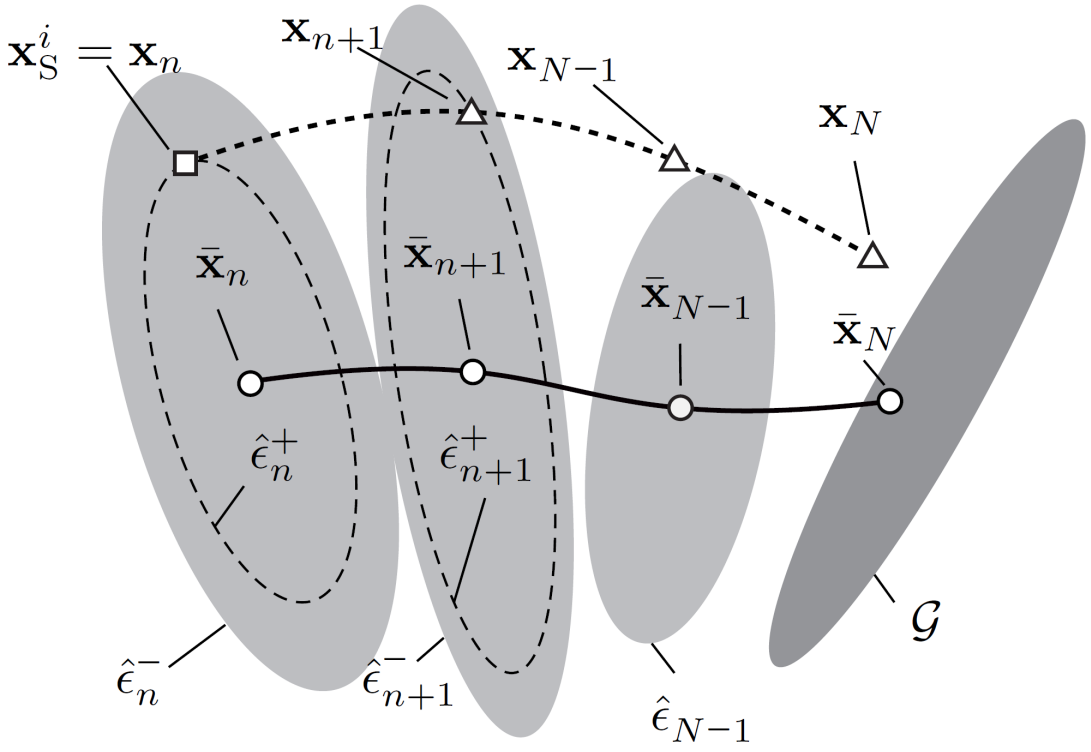


FIGURE 2.6: In this picture is shown how to adjsut the funnel hypothesis through simulation as it is proposed in [6]. Each ellipsoid represents a stabilizable ball to the goal with the action defined in each ball without violating the constraints. In this picture is shown that for the state $\mathbf{x}_s^i$ the final goal is not reached. Therefore it means that the funnel hypothesis is wrong and each ball has to be shrunken to integrate this new information from the simulation.

feedback structure and can be easily extended to multi-contacts scenarios.

Chapter 3

The Method

In this chapter we introduce and describe in detail the method that it is at the core of this manuscript. The proposed algorithm exploits the synergies arising from the combination of classical torque control methods and machine learning procedures.The method is built around the idea that enhancing control strategy through black box optimization can bring great performance improvements. The framework that we propose in this manuscript presents several advantages over other state of the art methods. The proposed method is extremely flexible and can be used to find the solution to different movement generation problems. This is possible because we do not make any strong assumption on the structure of the problem that we want to solve. In this thesis we apply this method in two different contexts: the automatic task combination and the optimization of trajectory to find a solution for challenging problems. The method was applied both on fixed and mobile robots, proving its capabilities each time. The method provides optimal solutions in respect to pre-designed objective functions. The optimization method that is employed gives great freedom to the user in designing objective functions that can be both non-linear and non-differentiable. This is possible because we use a derivative free method that exploits repeated simulation of the same experiment in order to find an optimal solution. Moreover, the proposed method relies on an optimization framework that can be easily extended for constrained optimization. We do not allow for any relaxation of the constraints, but aim for a solution that may be suboptimal but has to strictly satisfy all the constraints of the problem. This is one of the cardinal features of our method: providing controllers that are both optimal and safe. This is crucial when it comes to deploying the controller on the real robot. Having a solution that satisfies all of the constraints minimizes the risk that are common when the solution is transferred from the simulation environment to the real robot. In this manuscript we propose a novel optimization procedure to overcome some limitations that are intrinsic to the derivative free approach that we used in the our previous work.

Here we propose a method that alleviates the slow optimal solution convergence rate of the optimization methods that we used previously in our work. We have collected some preliminary results on analytical functions that indicate that this novel optimization method converges faster than the previous one and can be used even for constrained optimization problems.

3.1 Stochastic Optimization Framework

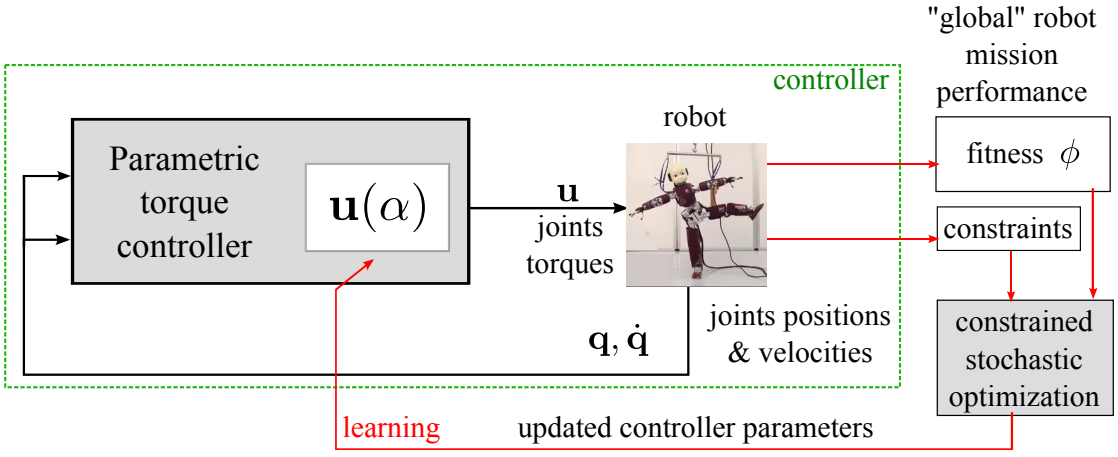


FIGURE 3.1: In this picture we show the general structure of the proposed framework. Our method is composed by two components: a parametrized controller and a constrained stochastic optimization routine. To optimize the controller parameters, we perform several repetitions of the experiment that we want to solve. The performance associated to the execution of each experiment is given as input to the constrained optimization algorithm, that updates the current parameters set toward the optimal solution

In this manuscript we propose a general framework to address several different robotics problems. The generality stems from the fact that we formalize our approach on very large assumptions while using universal tools like model based controllers and black box optimization methods. Our method is comprised of three main parts:

- a parametric torque controller
- a constrained stochastic optimizer
- a robot simulator

The model based torque controllers have a great advantage over the kinematic approaches. At the cost of greater modeling efforts, we can have a finer control strategy even for tasks defined in position and it allows us to explicitly model the forces that the robot exchanges with the environment. This is the only control framework that

can be used when the robot establishes multiple contacts with its environment and can give greater control for specific problems like robot balance. Even if kinematic control strategies for balance exist, all of them are employed as a means of planning strategy (like CoM trajectories planning and footstep placements according to those trajectories) and are not suited for fast real time compensation that is vital when the robot has to perform quick movements or when it has to hastily react to external disturbances. In the next chapter we showed how this kind of controller can be used for trajectory optimization for humanoid robots while keeping their balance. We decided to model our actor as a feedback controller instead of relying on any sort of learning technique (like neural networks *et similia*) in order to exploit the robustness and adaptability that characterize these kinds of approaches. We design the parameterization of the controller in a way that the structure of the controller is not altered, but we try to enhance it either by optimizing the input trajectory or introducing a weighting function in order to activate or deactivate each controller.

To optimize the free parameters, we introduce two elements, the fitness function and a set of inequality and equality constraints the fitness function computes a performance measure of the global task executed by the robot over T time steps with the current parameters. The fitness function can contain different criteria ranging from energy consumption arguments to specific properties of the desired trajectories (*e.g.* speed and smoothness). the constraints determine the admissible controls to be applied to the robot. They can be dependent on the robot structure (*e.g.* maximum joint torques and joint ranges), on the environment (*e.g.* obstacles and collisions), on the tasks (*e.g.* safety limits and couplings), *etc.*

In our work we decided to choose an optimization method that does not constrain the structure of the objective function and its differentiability property. For this reason we used CMA-ES [24] a derivative free method used for black box optimization. We chose this method over other approaches from the literature for many different reasons. CMA-ES needs only few parameters to be tuned in order to provide an optimal solution. CMA-ES because is an evolutionary strategy that can be easily extended for constrained optimization problems. Although CMA-ES shows a very slow converging rate, performing the optimization offline makes this issue less of a problem. The optimization loop is based on the repetition of the same experiments several times. All of the experiments are performed in simulation. Running the optimization inside a simulated environment it is very desirable, especially when want to tackle risky tasks like keeping the robot in balance or avoiding obstacles. In these cases, especially if we start the optimization from a random point, we will incur many failures that can lead to disastrous outcome than if they would have performed directly on the real robot. In our work we use different simulated environment. All the tasks that do not involve a force interaction are performed

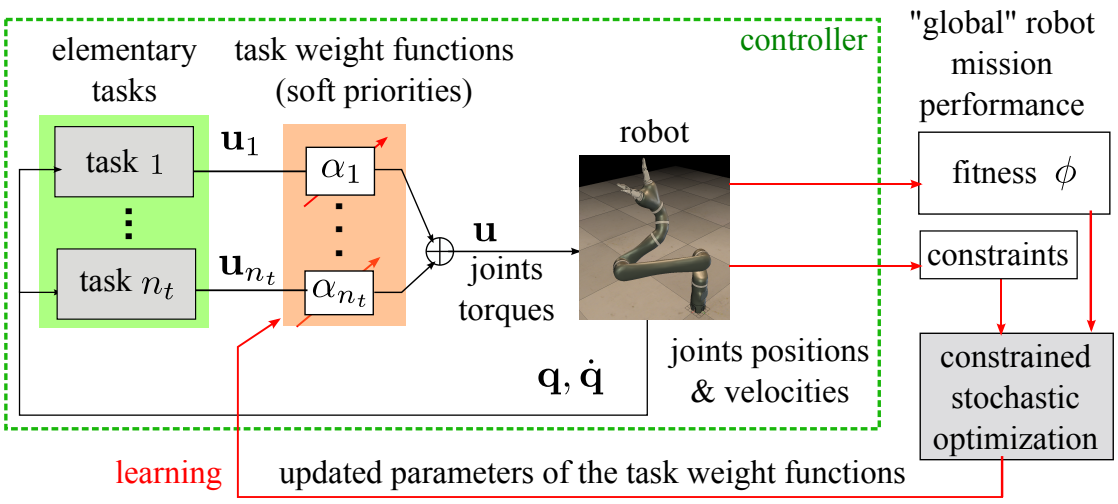


FIGURE 3.2: Overview of the proposed method. The controller consists of a weighted combination of elementary tasks, where the weight functions represent the soft task priorities. An outer learning loop enables the optimization of the task weight parameters, taking into account the constraint violations in an explicit way.

in Matlab with the aid of The Robotics Toolbox [56] and the mex-wholebodymodel library to provide a dynamical model of the robots. For the experiments that need a good simulation of the force interaction between the robot and the environment, we used Gazebo as simulator and we implemented our controller inside Simulink [57]. We integrate every simulation means with the control and optimization routine inside a set of Matlab libraries that are freely available in Github.

3.2 Learning Soft Task Priorities

Our method aims at automatically learning the task priorities (or task weight functions) to maximize the robot performance ensuring that the optimized priorities lead to behaviours that always satisfy the constraints. The global robot movement is evaluated by a fitness function ϕ that is used as a measure of the ability of the robot to fulfil its mission without violating the constraints. Our proposed method, outlined in Fig. 3.2, extends the framework that was introduced in [26]. In this section we recall the multi-tasks controller and the structure of the parametrized task weight functions α_i, while the optimization procedure is described in Section 3.4, where we analyze some recent extensions of the basic CMA-ES method that deal with constraints.

3.2.1 Controller for a Single Elementary Task

We hereby describe the torque controller for the i-th elementary task. To simplify the controller design, we decided to adopt the Unified Framework (UF) [21]. We consider

the well-known rigid-body dynamics of a robot with n DOF, *i.e.*,

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{u}_i(\mathbf{q}, \dot{\mathbf{q}}), \tag{3.1}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}} \in \mathbb{R}^n$ are, respectively, the joints positions, velocities and accelerations, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the generalized inertia matrix, $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^n$ accounts for Coriolis, centrifugal and gravitational forces and $\mathbf{u}_i(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^n$ is the vector of the commanded torques of the i-th task. Using the same notation as in [21], we describe the task as a constraint, given by:

$$\mathbf{h}_i(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{0}\ , \tag{3.2}$$

where $\mathbf{h}_i \in \mathbb{R}^m$ is at least a twice differentiable function, where m is the task dimension. By differentiating the constraint with respect to time, we obtain:

$$\mathbf{A}_i(\mathbf{q}, \dot{\mathbf{q}}, t)\ddot{\mathbf{q}} = \mathbf{b}_i(\mathbf{q}, \dot{\mathbf{q}}, t), \tag{3.3}$$

where $\mathbf{A}_i(\mathbf{q}, \dot{\mathbf{q}}, t)$ is a known $m \times n$ matrix and $\mathbf{b}_i(\mathbf{q}, \dot{\mathbf{q}}, t)$ is a $m \times 1$ vector. For example, given a simple tracking control task with $\mathbf{h}_i(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{q} - \mathbf{q}^{des}$, where $\mathbf{q}^{des}$ corresponds to a desired trajectory. By computing the second order derivative in t we obtain $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}^{des}$, where $\mathbf{b} = \ddot{\mathbf{q}}^{des}$ and $\mathbf{A}_i = \mathbf{I}$ (with $\mathbf{I}$ the identity matrix). Applying Gauss's principle, it is possible to derive a controller that fulfils the constraints by minimizing the cost function $\zeta_i(t) = \mathbf{u}_i^\top \mathbf{N}_i(t)\mathbf{u}_i$, where $\mathbf{N}_i(t)$ is a positive semidefinite matrix. The optimization problem is defined by

$$\mathbf{u}_i^* = \underset{\mathbf{u}_i}{\operatorname{argmin}}\ \zeta_i(t) = \underset{\mathbf{u}_i}{\operatorname{argmin}}\ [\mathbf{u}_i^\top \mathbf{N}_i(t)\mathbf{u}_i]\,, \tag{3.4}$$

subject to Eq. 3.1 and 3.3. The solution to this optimization problem is given by

$$\mathbf{u}_i = \mathbf{N}_i^{-\frac{1}{2}}(\mathbf{A}_i\mathbf{M}^{-1}\mathbf{N}_i^{-\frac{1}{2}})^{\#}(\mathbf{b}_i + \mathbf{A}_i\mathbf{M}^{-1}\mathbf{f}), \tag{3.5}$$

where $(\cdot)^{\#}$ is the Moore-Penrose pseudoinverse and the upper script in $\mathbf{N}_i^{-\frac{1}{2}}$ denotes the inverse of the matrix square root. Controllers derived from UF are sensitive to kinematic singularities, due to the matrix inversion [58]. To overcome this problem, we reformulate the UF controller in a *regularized* fashion, as classically done at the kinematic level, for instance in [59]. The objective function of UF can be reformulated in such a way that the solutions of the optimization problem naturally exhibit a damped least squares structure (at the price of a loss of precision in the execution of the elementary task). Given the dynamical model of the robot (Eq. 3.1) and the constraint that describes the task (Eq.

3.3), we define the regularized optimal control problem:

$$\underset{\mathbf{u}_i}{\operatorname{argmin}}\, \zeta_i(t) = \underset{\mathbf{u}_i}{\operatorname{argmin}} \left[(\mathbf{A}_i \ddot{\mathbf{q}} - \mathbf{b}_i)^2 + \mathbf{u}_i^\top \frac{\mathbf{N}_i(t)}{\lambda_i} \mathbf{u}_i \right], \tag{3.6}$$

where λ_i is the regularizing factor with a l^2-weighted norm for the regularization term. In the simplest case, λ_i can be a manually-tuned constant value, or automatically determined by more sophisticated methods, as done in [60], based on the smallest singular value of the matrix to invert. To derive the closed form solution of the optimization problem, we substitute $\ddot{\mathbf{q}}$ in Eq. 3.6 with the expression obtained by solving the dynamical constraint for $\ddot{\mathbf{q}}$. The resulting closed form solution of the controller for a single elementary task is then:

$$\mathbf{u}_i = \mathbf{N}_i^{-1} \tilde{\mathbf{M}_i}^\top (\mathbf{I}\lambda_i^{-1} + \tilde{\mathbf{M}_i} \mathbf{N}_i^{-1} \tilde{\mathbf{M}_i}^\top)^{-1} (\mathbf{b}_i + \tilde{\mathbf{M}_i} \mathbf{f}), \tag{3.7}$$

with $\tilde{\mathbf{M}_i} = \mathbf{A}_i \mathbf{M}^{-1}$.

3.2.2 Controller for Multiple Elementary Tasks with Soft Task Priorities

With reference to the scheme of Fig. 3.2, we consider a number n_t of elementary tasks, that can be combined by the robot to accomplish a given "global" mission. The solution of the i-th task is provided by the torque controller $\mathbf{u}_i$ described in the previous section. Each task is modulated by a task priority or task weight function $\alpha_i(t)$. The ensemble $\{\alpha_i(t)\}_{i=1,\dots,n_t}$ constitutes the *activation policy* that determines the overall robot movement. The robot controller is therefore given by

$$\mathbf{u}(\mathbf{q}, \dot{\mathbf{q}}, t) = \sum_{i=1}^{n_t} \alpha_i(t)\, \mathbf{u}_i(\mathbf{q}, \dot{\mathbf{q}}) \ , \tag{3.8}$$

where t is the time, and $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the robot joint positions and velocities. The task priorities $\alpha_i(t)$ are scalar functions and their time profile can be optimized. We automatically tune the task priorities with a learning algorithm. We seek the best task weight functions that maximize a defined performance measure, or fitness, evaluating the execution of the global task. As finding the optimal functions $\alpha_i^*(t)$ is an intractable problem, we turn the functional optimization problem into a numerical optimization problem by representing the task priorities with parametrized functional approximators, $\alpha_i(t) \to \hat{\alpha}_i(\boldsymbol{\pi}_i, t)$, where $\boldsymbol{\pi}_i$ is the set of parameters that shape the temporal profile of

the i-th task weight function. The controller then becomes:

$$\mathbf{u}(\mathbf{q}, \dot{\mathbf{q}}, t) = \sum_{i=1}^{n_t} \hat{\alpha}_i(\boldsymbol{\pi}_i, t)\, \mathbf{u}_i(\mathbf{q}, \dot{\mathbf{q}}) \tag{3.9}$$

Finding the optimal task priorities consists therefore in finding the optimal parameters $\boldsymbol{\pi}_i^*$, which can be done applying a learning method to maximize the fitness ϕ.

3.2.3 Learning the Task Priorities

We model the task priorities as a weighted sum of normalized Radial Basis Functions (RBFs):

$$\hat{\alpha}_i(\hat{\boldsymbol{\pi}}_i, t) = S\left(\frac{\sum_{k=1}^{n_r} \hat{\pi}_{ik}\psi_k(\mu_k, \sigma_k, t)}{\sum_{k=1}^{n_r} \psi_k(\mu_k, \sigma_k, t)}\right), \tag{3.10}$$

where $\psi_k(\mu_k, \sigma_k, t) = \exp\left(-1/2[(t-\mu_k)/\sigma_k]^2\right)$, with fixed mean μ_k and variance σ_k of the basis functions, n_r is the number of RBFs and $\hat{\boldsymbol{\pi}}_i = (\hat{\pi}_{i1}, \ldots, \hat{\pi}_{in_r}) \subseteq \mathbb{R}^{n_P}$ is the set of parameters for each task priority. $S(\cdot)$ is a sigmoid function that squashes the output to the range $[0, 1]$. The elementary task is fully activated when the task priority is equal to 1, otherwise the control action fades out until a full deactivation occurs when the priority goes to 0. The free parameters $\hat{\boldsymbol{\pi}}_i$ of each task weight function (Eq. 3.10) constitute the current parameters set to optimize: $\boldsymbol{\pi} = (\hat{\boldsymbol{\pi}}_1, \ldots, \hat{\boldsymbol{\pi}}_{n_t})$.

3.3 Learning Optimal Trajectories

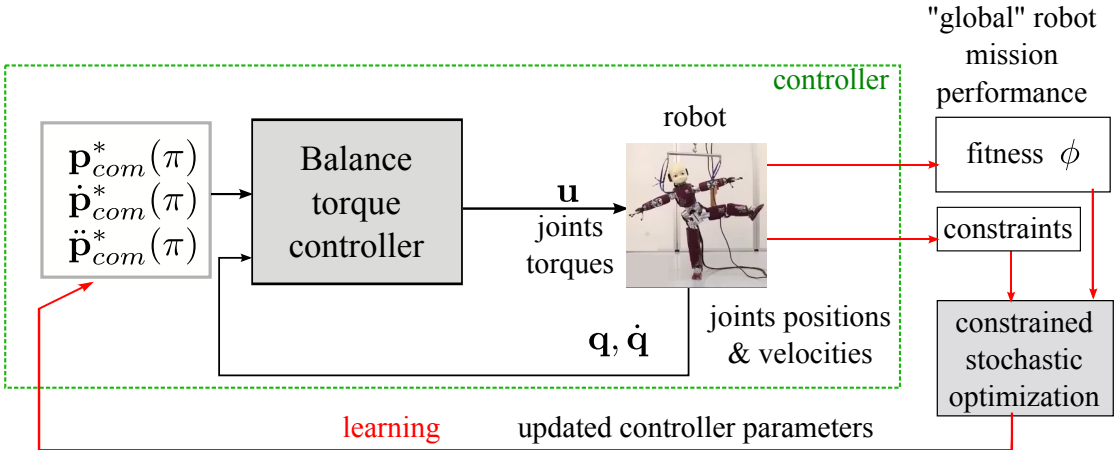


FIGURE 3.3: Overview of the proposed method. The controller consists of a weighted combination of elementary tasks, where the weight functions represent the soft task priorities. An outer learning loop enables the optimization of the task weight parameters, taking into account the constraint violations in an explicit way.

Inspired by the idea that by combining machine learning and control theory we can obtain better results, we design a learning control system where the controller is enhanced through an iterative learning procedure. To show the great capabilities of this approach we decide to face with the problem of balance for humanoids robot for highly dynamic task. The structure of the proposed framework is rather simple: we introduce a parametric reference for the torque balance controller of the humanoid robot. Than we perform a complete experiment and we evaluate the fitness of the current parameter set. The cost to go of the current roll-out is provided as input to an instance of constrained (1+1)CMA-ES [7], then a new parameter set is computed and tested by repeating the same experiment in simulation. This approach does not request the intervention of an expert to design a feasible trajectory and minimize the effort to deploy the solution on the humanoids platform in use. On the other side this method inherits from the feedback controller the robustness of a closed loop solution and the possibility to extend the current framework to an ample set of multi-contact scenario. In the next section we will refer to [61], [23] to introduce and describe the balance controller in use.

3.3.1 Balance Controller

Balance control plays a major role when it comes to human robot cooperation. We need reliable controllers that are robust under a wide range of disturbances.If for now we ignore the friction, from a physical perspective, a stance is in balance when the Center of Pressure (CoP) stays inside the area of the feet in contact with the ground (commonly called the support polygon) . The CoP represents the application point of the Ground Reaction Force (GRF), and they represents all the forces acting between the the robot feet and the supporting surface. To the extent of our knowledge, results from literature [62] [63] show that controlling both angular and linear momentum of the floating base platform is responsible for a full control of the CoP of the robot stance. The D'Alambert principle describes the dynamical relations between the external forces acting on the robot and the rate of change of the linear and angular momentum of the robot in its Center of Mass (CoM).

$$\dot{\mathbf{l}} = m\mathbf{g} + \mathbf{w} \tag{3.11}$$

$$\dot{\mathbf{k}} = (\mathbf{p} - \mathbf{r}_G) \times \mathbf{w} + \boldsymbol{\tau}_n \tag{3.12}$$

$$\tag{3.13}$$

where $\mathbf{r}_G$ is the center of mass position, $\mathbf{p}$ is the CoP position, $\mathbf{w}$ is the force component of the external wrench $\mathbf{f} = [\mathbf{w}, \boldsymbol{\tau}] \in \mathbb{R}^6$, $\boldsymbol{\tau}_n$ is the component of the external wrench torque orthogonal to the ground and together $\mathbf{k}$ and $\mathbf{l}$ (respectively the rotational and

linear momentum) for the spatial centroidal momentum $\mathbf{h} = [\mathbf{l}, \mathbf{k}]^T \in \mathbb{R}^6$. The spatial centroidal momentum represents the summation of all the linear and angular momentum of each link of the robot projected in a reference frame that is centred in the instantaneous center of mass. Due to the relation in 3.11 to achieve a full control of the CoP we need to act both on the rate of change of the linear and the angular momentum. Given the latter considerations, momentum-based balance controllers represents to our knowledge the best means to manage the robotic balance control issue.
We hereby describe the balance torque controller. Let us consider a dynamic representation of a floating base robot. We represent the configuration space of a humanoid robot by a triplet $\mathbf{q} = ({}^I\mathbf{p}_B, {}^IR_B, \mathbf{q}_j) \in \mathbb{R}^3 \times SO(3) \times \mathbb{R}^n$ where ${}^I\mathbf{p}_B$ and ${}^IR_B$ describe respectively the position and orientation of the base of the humanoid (usually located in the torso) respect of an inertial reference frame, while $\mathbf{q}_j$ represents the vector of joint angles. The total Degrees of Freedom (DoF) of the robot are $n+6$ because the free floating base adds 6 DoF. The generalized velocity associated to the system is $\boldsymbol{\nu} = ({}^I\dot{\mathbf{p}}_B, {}^I\boldsymbol{\omega}_B, \dot{\mathbf{q}}_\mathbf{j}) = (\mathbf{v}_B, \dot{\mathbf{q}}_j)$ where $\mathbf{v}_B$ describes the spatial velocities of the base frame with respect to the inertial frame; ${}^I\boldsymbol{\omega}_B$ is defined as $[{}^I\boldsymbol{\omega}_B]_\times = {}^I\dot{R}_B {}^IR_B^T$, with $[\cdot]_\times$ being the screw symmetric matrix obtained from ${}^I\boldsymbol{\omega}_B$. With the hypothesis of a flat contact surface (reasonable hypothesis in our case, since our contacts are on a flat ground and flat chair) the constrained dynamic model describing the robot is:

$$0 = \mathbf{J}_{C_k}\boldsymbol{\nu} \tag{3.14}$$

$$\mathbf{Su} = \mathbf{M}(\mathbf{q})\dot{\boldsymbol{\nu}} + \mathbf{C}(\mathbf{q}, \boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{G}(\mathbf{q}) - \sum_k \mathbf{J}_{C_k}\mathbf{f}_k \tag{3.15}$$

where $\mathbf{u} \in \mathbb{R}^n$ denotes the generalized force at the joints, $\mathbf{S}$ is a selection matrix, $\mathbf{M} \in \mathbb{R}^{n+6} \times \mathbb{R}^{n+6}$ is the generalized mass matrix, $\mathbf{C}(\mathbf{q}, \boldsymbol{\nu})$ and $\mathbf{G}(\mathbf{q})$ account respectively for the non inertial forces and the gravity torque. $\mathbf{f}_k \in \mathbb{R}^6$ represents the external wrench and is related to the Ground Reaction Forces (GRF) and the Center of Pressure (CoP) while the Jacobian $\mathbf{J}_{C_k}$ maps the robot velocity to the linear and angular velocity of the frame where the external wrench is exerted:

$$\mathbf{J}_{C_k}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}^b_{C_k}(\mathbf{q}) & J^j_{C_k}(\mathbf{q}) \end{bmatrix} \tag{3.16}$$

$$\mathbf{J}^b_{C_k} = \begin{bmatrix} \mathbf{1}_3 & -[{}^I\mathbf{p}_{C_k} - {}^I\mathbf{p}_B]_\times \\ \mathbf{0}_3 & \mathbf{1}_3 \end{bmatrix} \tag{3.17}$$

The dimension of $\mathbf{J}_{C_k}$ depends on the number and the kind of contacts with the environment. For example a double stance contact that constraints both orientation and position of the contact point has 12 dimensions (6 for each foot in contact with the ground). Equation 3.14 is equal to zero because we make the hypothesis of no slippage on the contact surfaces. Due to the fact that a free floating model is underactuated, we

can rewrite the equation 3.15 into two different parts:

$$\mathbf{0} = \mathbf{J}_{C_k}\dot{\boldsymbol{\nu}} + \dot{\mathbf{J}}_{C_k}\boldsymbol{\nu} \tag{3.18}$$

$$\mathbf{0} = \begin{bmatrix}\mathbf{M}_b(\mathbf{q}) & \mathbf{M}_{bj}(\mathbf{q})\end{bmatrix}\dot{\boldsymbol{\nu}} + \mathbf{C}_b(\mathbf{q},\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{G}_b(\mathbf{q}) - \textstyle\sum_k \mathbf{J}^b_{C_k}\mathbf{f}_k \;\in \mathbb{R}^6 \tag{3.19}$$

$$\mathbf{u} = \begin{bmatrix}\mathbf{M}_{bj}(\mathbf{q})^T & \mathbf{M}_j(\mathbf{q})\end{bmatrix}\dot{\boldsymbol{\nu}} + \mathbf{C}_j(\mathbf{q},\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{G}_j(\mathbf{q}) - \textstyle\sum_k \mathbf{J}^j_{C_k}\mathbf{f}_k \;\in \mathbb{R}^n \tag{3.20}$$

with $\mathbf{M}_b(\mathbf{q}) \in \mathbb{R}^{6\times 6}$ $M_j(\mathbf{q}) \in \mathbb{R}^{n\times n}$ and $\mathbf{M}_{bj}(\mathbf{q}) \in \mathbb{R}^{6\times n}$. It is proven that it is possible to diagonalize the generalized mass matrix $\mathbf{M}$ leading to a decoupling of base and joint accelerations [64]. We introduce the following change of coordinates $\mathbf{q} = \mathbf{q}$, $\overline{\boldsymbol{\nu}} = \mathbf{T}(\mathbf{q})\boldsymbol{\nu}$

$$\mathbf{T} = \begin{bmatrix}{}^c\mathbf{X}_B & {}^c\mathbf{X}_B\mathbf{M}_b^{-1}\mathbf{M}_{bj} \\ \mathbf{0}_{n\times 6} & \mathbf{1}_n\end{bmatrix} \tag{3.21}$$

$${}^c\mathbf{X}_B = \begin{bmatrix}\mathbf{1}_3 & -[{}^I\mathbf{p}_c - {}^I\mathbf{p}_B]_\times \\ \mathbf{0}_{3\times 3} & \mathbf{1}_3\end{bmatrix} \tag{3.22}$$

where the c superscript represents the frame centered in the center of mass of the floating base platform, oriented as the inertial frame I. Equation 3.15 becomes:

$$\mathbf{S}\mathbf{u} = \overline{\mathbf{M}}(\mathbf{q})\dot{\overline{\boldsymbol{\nu}}} + \overline{\mathbf{C}}(\mathbf{q},\overline{\boldsymbol{\nu}})\overline{\boldsymbol{\nu}} + \overline{\mathbf{G}} - \sum_i \mathbf{J}^T_{C_i}\mathbf{f}_i \tag{3.23}$$

where

$$\overline{\mathbf{M}} = \begin{bmatrix}\overline{\mathbf{M}}_b(\mathbf{q}) & \mathbf{0}_{6\times n} \\ \mathbf{0}_{n\times 6} & \overline{\mathbf{M}}_j(\mathbf{q}_j)\end{bmatrix} \tag{3.24}$$

$$\overline{\mathbf{M}}_b(\mathbf{q}) = \begin{bmatrix}m\mathbf{1}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \tilde{\mathbf{I}}(\mathbf{q})\end{bmatrix} \tag{3.25}$$

$$\overline{\mathbf{G}} = mg\,\mathbf{e}_3 \tag{3.26}$$

$$\overline{\mathbf{J}}_{C_i}(\mathbf{q}) = \begin{bmatrix}\overline{\mathbf{J}}^b_{C_i}(\mathbf{q}) & \overline{\mathbf{J}}^j_{C_i}(\mathbf{q})\end{bmatrix} \tag{3.27}$$

$$\overline{\mathbf{J}}^b_{C_i} = \begin{bmatrix}\mathbf{1}_3 & -[{}^I\mathbf{p}_{C_i} - {}^I\mathbf{p}_c]_\times \\ \mathbf{0}_{3\times 3} & \mathbf{1}_3\end{bmatrix} \tag{3.28}$$

with m the mass of the robot and $\tilde{\mathbf{I}}$ the inertia of the robot defined in the CoM frame. Due to the change of coordinate, the floating base velocities now represent the linear and angular velocities of the CoM. In particular the angular velocity is the *average angular velocity*. Equation 3.23 unifies in one expression the centroidal dynamics and the dynamic description of the free floating base. From now on all the subsequent equations will refer to Equation 3.23 and to lighten the notation we will drop the overline sign. The change of coordinate allows for a simple computation of the robot momentum. From

[61] and Equation 3.23,the robot momentum is defined as $\mathbf{h} = \mathbf{M}_b\mathbf{v}_b$.The D'Alembert's principle states that the rate of change of the robot momentum depends on the external forces acting on the robot (force from the environment and gravity in the CoM). From Equation 3.23 and d'Alembert's principle we can write:

$$\dot{\mathbf{h}} = \frac{d}{dt}(\mathbf{M}_b\mathbf{v}_B) = \mathbf{J}_B^T\mathbf{f} - mg\ \mathbf{e}_3 \tag{3.29}$$

By inverting the previous relation we can find an expression that connects the desired rate of the change of the centroidal momentum and the external forces:

$$\mathbf{f} = \mathbf{J}_B^{-T}(\dot{\mathbf{h}}^* + mg\ \mathbf{e}_3) \tag{3.30}$$

$$\dot{\mathbf{h}}^* = \dot{\mathbf{h}}^{des} + \mathbf{K}_p(\mathbf{h} - \mathbf{h}^d) + \mathbf{K}_i \int \mathbf{h} - \mathbf{h}^d \tag{3.31}$$

where $\dot{\mathbf{h}}^*$ combines a feed-forward term and a Proportional Integral correction to ensure that the system achieves the desired centroidal momentum. The control torques that realize the instantaneous external forces and satisfy the holonomic constraints at the contact point are:

$$\mathbf{u} = \mathbf{\Lambda}^\dagger(\mathbf{J}_C\mathbf{M}(\mathbf{F} - \mathbf{J}_C^T\mathbf{f}) - \dot{\mathbf{J}_C}\boldsymbol{\nu})) + \mathbf{N}_\Lambda\mathbf{u}_0 \tag{3.32}$$

with $\mathbf{\Lambda} = \mathbf{J}_j\mathbf{M}_j$, $(\cdot)^\dagger$ the pseudo-inversion, $\mathbf{F} = \mathbf{C} + \mathbf{G}$ and $\mathbf{u}_0 = \mathbf{F} - \mathbf{J}_j\mathbf{f} + \mathbf{K}_p^j(\mathbf{q}_j - \mathbf{q}_j^{des}) + \mathbf{K}_d\dot{\mathbf{q}}_\mathbf{j}$ the posture task projected in the null space of the primary task that is introduced for the stability of the zero dynamics. In [22] the authors empirically proved that in a centroidal momentum controller the posture task do not always assure the internal stability. To overcome this issue in Equation 3.31 they replace the momentum error with

$$\mathbf{h_{err}} = \begin{bmatrix} \overline{\mathbf{J}}_G^l(\mathbf{q}_j) \\ \overline{\mathbf{J}}_G^k(\mathbf{q}_j^d) \end{bmatrix} \dot{\mathbf{q}}_j \tag{3.33}$$

where $[\overline{\mathbf{J}}_G^l(\mathbf{q}_j)\overline{\mathbf{J}}_G^k(\mathbf{q}_j^d)]$ comes from a reduced expression of the centroidal momentum matrix that maps the joints velocities to the velocities of the floating base and is defined as $\overline{\mathbf{J}}_G(\mathbf{q}_j) = -\mathbf{M}_b\mathbf{J}_b^{-1}\mathbf{J}_j$. This substitution is motivated by the Lyapunov stability of the linearized closed loop dynamics around the joint positions-velocities equilibrium point $(\mathbf{q}_j^d, 0)$.

Equation 3.30 is mathematically sound only if the robot is standing on one foot and we ignore all the balance related constraints (GRF contained inside the friction cone and CoP inside the support polygon). To resolve the redundancy in the external forces in a multi-contact scenario and to take into account all the conditions for a stable stance we can recast the computation of the external forces as an optimization problem. This is done in our case, since the robot transitions between 2 contact forces in the legs when

seated (contacts between legs and chair) and 2 in the feet when it stands up (contacts between feet sole and ground).

3.3.2 Trajectories Parametrization

for this application the cartesian trajectory is modelled as a weighted sum of normalized Radial Basis Functions (RBFs):

$$\hat{\alpha}_i(\hat{\boldsymbol{\pi}}_i, t) = \frac{\sum_{k=1}^{n_r} \hat{\pi}_{ik} \psi_k(\mu_k, \sigma_k, t)}{\sum_{k=1}^{n_r} \psi_k(\mu_k, \sigma_k, t)} \tag{3.34}$$

where $\psi_k(\mu_k, \sigma_k, t) = \exp\left(-1/2[(t-\mu_k)/\sigma_k]^2\right)$, with fixed mean μ_k and variance σ_k of the basis functions, n_r is the number of RBFs and $\hat{\boldsymbol{\pi}}_i = (\hat{\pi}_{i1}, \ldots, \hat{\pi}_{in_r}) \subseteq \mathbb{R}^{n_P}$ is the set of parameters for each task priority. Generally for the design of cartesian trajectory some conditions are needed to be meet for example passing through a set waypoints or imposing particular conditions on initial or final velocities and accelerations to cite a few. In this work, due to the application scenario, we employed an extend version of the RBF that allows for waypoints. We augment the RBF model by adding a set of RBFs ψ_j, one for each waypoint, located in correspondence of the time μ_j in which we impose the waypoint passage:

$$\hat{\alpha}_i(\hat{\boldsymbol{\pi}}_i, t) = \frac{\sum_{j=1}^{n_{wp}} \hat{\pi}^c_{ij} \psi_j(\mu_j, \sigma_j, t) + \sum_{k=1}^{n_r} \hat{\pi}^f_{ik} \psi_k(\mu_k, \sigma_k, t)}{\sum_{k=1}^{n_r+n_{wp}} \psi_k(\mu_k, \sigma_k, t)} \tag{3.35}$$

Once we set the value of the free parameters $\hat{\boldsymbol{\pi}}^f_i$, we need to compute the value of the fixed parameters in order to match the waypoint passage conditions. Given the vector of fixed value $\mathbf{f^{wp}} = [\mathbf{f1}, \ldots, \mathbf{f_{n_{wp}}}]$ we can write:

$$\hat{\boldsymbol{\pi}}_i{}^c = A^{-1} b \tag{3.36}$$

where

$$A = \begin{bmatrix} \psi_1(\mu^c_1, \sigma_1, \mu^c_1) & \cdots & \psi_{n_{wp}}(\mu^c_{n_{wp}}, \sigma_{n_{wp}}, \mu^c_1) \\ \vdots & & \vdots \\ \psi_1(\mu^c_1, \sigma_1, \mu^c_{n_{wp}}) & \cdots & \psi_{n_{wp}}(\mu^c_{n_{wp}}, \sigma_{n_{wp}}, \mu^c_{n_{wp}}) \end{bmatrix} \tag{3.37}$$

$$b = \begin{bmatrix} f^{wp}_1 \sum_{k=1}^{n_r} \hat{\pi}^f_{ik} \psi_k(\mu_k, \sigma_k, \mu^c_1) - \sum_{k=1}^{n_r+n_{wp}} \psi_k(\mu_k, \sigma_k, \mu^c_1) \\ \vdots \\ f^{wp}_{n_{wp}} \sum_{k=1}^{n_r} \hat{\pi}^f_{ik} \psi_k(\mu_k, \sigma_k, \mu^c_{n_{wp}}) - \sum_{k=1}^{n_r+n_{wp}} \psi_k(\mu_k, \sigma_k, \mu^c_{n_{wp}}) \end{bmatrix} \tag{3.38}$$

For the case of exponential radial basis functions the matrix A is always invertible due to its particular structure.

3.4 Black-Box Optimization

In the context of global optimization is not unusual that the objective function is considered as a black box: in such case we ignore the analytical expression and we are not capable of computing its derivative in closed form. In particular the differentiability properties of the objective function determines which approach has to be employed in order to find a solution for the optimization problem. If the objective function is differentiable with respect to the controls and the parameters (which requires the function approximators to be differentiable as well with respect to the controls [60]), then gradient methods can be used. If the fitness is not differentiable with respect to the parameters, then a derivative-free method can be used. Usually in the context of black box optimization we can access the value of our objective function only using sparse query input points which give us noisy response. Black box optimization requires that each dimension of the parameter space that where we want to find an optimal solution, is bounded. So is reasonable to suppose that our optimization problem is confined inside an hyperrectangle of the same dimension of the search space. In our method, the learning procedure is an attempt to find an optimal solution for the free parameters vector $\boldsymbol{\pi}$ without knowing the real structure of the objective function. Given T time steps, the fitness function is defined as $\phi = \phi(\mathbf{q}_{t=1,\dots,T}, \mathbf{u}_{t=1,\dots,T}, t)$. ϕ describes the performance of the controller in fulfilling its global mission. The objective function could be a simple measure of success in case of goal reaching, the time duration of a movement, the energy consumption *etc.* More criteria for optimizing robot motions in optimal control frameworks are reported in [65]. Derivative-free methods are appealing, since they do not constrain the design of the objective function. Furthermore, recent results showed that it is possible to achieve very fast performances in trial-and-error learning with derivative-free methods [66]. The black box optimization problems that we want to solve are formalized in the following way. Given an objective function $\phi(\boldsymbol{\pi}) : \mathbb{R}^{n_P} \to \mathbb{R}$ and a set of equality and inequality constraints g, h, we want to find an optimal solution $\boldsymbol{\pi} \in \boldsymbol{\Pi} \subseteq \mathbb{R}^{n_P}$ to the problem:

$$\boldsymbol{\pi}^{\circ} = \arg\max_{\boldsymbol{\pi}} \phi(\boldsymbol{\pi}) \tag{3.39}$$

s.t.

$$g_i(\boldsymbol{\pi}) \leq 0 \quad i = 1, \dots, n_{IC} \tag{3.40}$$

$$h_i(\boldsymbol{\pi}) = 0 \quad i = 1, \dots, n_{EC} \tag{3.41}$$

where $g_i(\boldsymbol{\pi})$ with $i = 1, \dots, (\boldsymbol{\pi})$ represents all the inequality constraints and $h_i(\boldsymbol{\pi})$ with $i = 1, \dots, n_{EC}$ describes all the equality constraints.

CMA-ES without constraints

function CMA-ES
  for each $gen = 1, \ldots, n_{GENERATIONS}$ **do**
    for each $k = 1, \ldots, K$ **do**
      $\boldsymbol{\pi}_k \sim \mathcal{N}(\bar{\boldsymbol{\pi}}, \varsigma^2 \boldsymbol{\Sigma})$ ▷ **samples**
      $\phi_k = \phi(\boldsymbol{\pi}_k)$ ▷ **evaluation**
    end for
    $\boldsymbol{\pi}_{1:K_e} = \textsc{Sort}(\boldsymbol{\pi}_{k=1:K}, J_{k=1:K})$ ▷ **sorting**
    $\bar{\boldsymbol{\pi}}^{new} = \sum_{k=1}^{K_e} P_k \boldsymbol{\pi}_k$ with $\sum_{k=1}^{K_e} P_k = 1$
    $\boldsymbol{\Sigma}^{new} = \textsc{UpdCov}(\bar{\boldsymbol{\pi}}^{new}, P_{k=1:K_e})$
    $\varsigma^{new} = \textsc{UpdStepSize}(\varsigma)$
    $\bar{\boldsymbol{\pi}} = \bar{\boldsymbol{\pi}}^{new} \quad \boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{new} \quad \sigma = \sigma^{new}$
  end for
end function

FIGURE 3.4: Pseudocode for the basic CMA-ES without constraints.

3.5 CMA-ES

CMA-ES [24] is a stochastic derivative-free optimization strategy that is suitable for non-linear and non-convex objective functions. This method belongs to the class of evolutionary algorithms. CMA-ES algorithm exploits two main principles in order to update the parameters of the search distribution to find an optimal solution.

the first principle consist in increasing the probability of discovering new successful candidate through the application of a maximum-likelihood approach. Mean and covariance updates act as a natural gradient descent. The idea behind mean distribution update is that the algorithm tries to maximize the likelihood of the last successful sample while the covariance matrix update increases the likelihood of previous search steps that proved to ameliorate the fitness. The Covariance Matrix Adaptation incorporates the idea of a principal components analysis without discarding any principal axes. Other evolutionary algorithm that employs search distribution like Cross-Entropy Method [67] make use of very similar ideas, but the covariance matrix update is based on successful points and they not take into account the direction of the improvement. The second principle is based on the idea that exploiting the information among consecutive steps is beneficial for the optimization. two paths store the evolution of the distribution mean throughout the course of all the past generations. We refer to this path as *search* or *evolution* paths. These paths try to capture information about successful consecutive steps. When the evolution paths become long it means that we have just got several consecutive successful steps. The evolution paths provides different information about the evolution of the search distribution. One path accelerates the peace at witch the covariance matrix is updated to set a preference for successful update directions. The other path fulfils the function of step-size control. In this way the second path tries

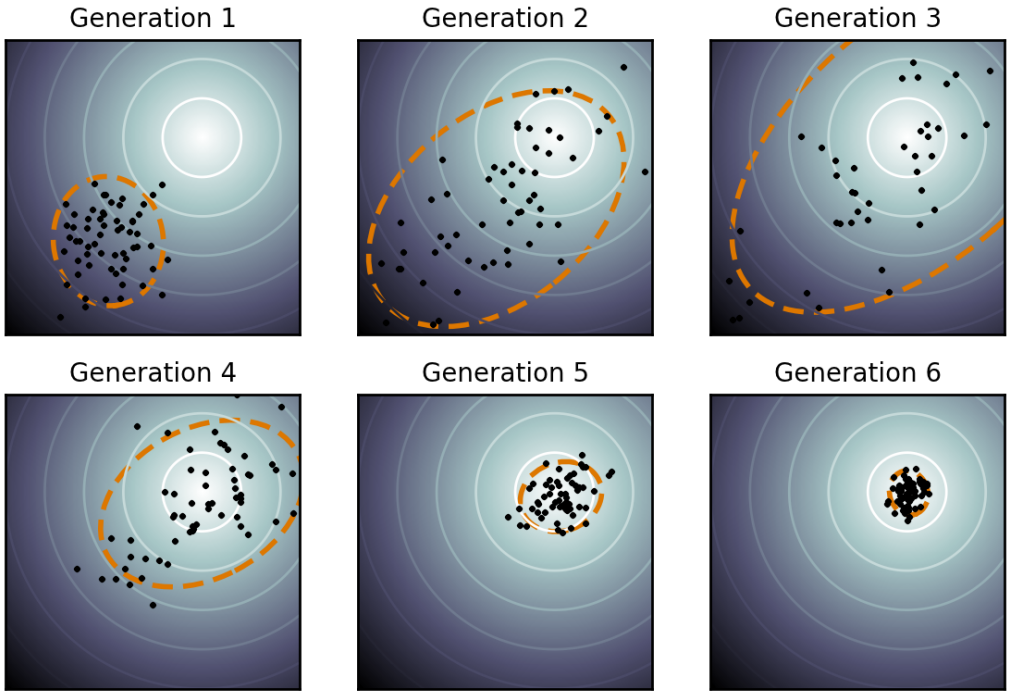


FIGURE 3.5: This image shows the evolution in time of the search distribution toward the optimum for a simple two dimensional problem. The contour lines in the picture represents the region of the space where objective function presents the same value while the brighter area represent portion of the space where the objective function reaches the highest values. The sequence of images shows that, after six generation, the search distribution has reached the maximum. One of the most interesting feature of CMA-ES is the capability to adapt the step-size exploiting the information collected during the optimization process. Source: https://en.wikipedia.org/wiki/CMA-ES

to prevent premature convergence while keeping a good converge rate to the optimum. In the following we outline the most commonly implementation of CMA-ES. At each iteration of the algorithm, new candidate solutions are generated from a population of candidates through stochastic sampling. The fitness function is then used to evaluate the candidates. In our case, each candidate is a possible set of parameters for the task that we want to solve $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_{n_P}\}$. At each iteration of the algorithm (called *generation*), a new population of candidates is generated by sampling a multivariate normal distribution $\mathcal{N}(\bar{\boldsymbol{\pi}}, \boldsymbol{\Sigma})$, where $\bar{\boldsymbol{\pi}}$ and $\boldsymbol{\Sigma}$ represent respectively mean and covariance of the distribution. A fitness value is computed for each candidate in the current population and, according to the fitness, only the most promising candidates are kept to update the search distribution. Given K candidates $\{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K\}$, the algorithm selects the $K_e < K$ best ones according to their ordered fitness values $\{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{K_e}\}$. It uses the selected candidates to compute the mean of the sampling distribution at the next iteration: $\bar{\boldsymbol{\pi}}^{(new)} = \sum_{i=1}^{K_e} P_i \boldsymbol{\pi}_i$, with $\sum_{i=1}^{K_e} P_i = 1$.

Then the covariance matrix is updated as:

$$\mathbf{\Sigma}^{(new)} = (1 - c_1 - c_2)\mathbf{\Sigma} + c_1 \mathbf{p}_\Sigma \mathbf{p}_\Sigma^T + c_2 \sum_{i=1}^{K_e} P_i \mathbf{y}_i \left(\mathbf{y}_i\right)^T$$

with $\mathbf{y}_i = (\boldsymbol{\pi}_i - \bar{\boldsymbol{\pi}})$, c_1 and c_2 two predefined parameters (see [24] for more details). The symbol $\mathbf{p}_\Sigma$ is a term measuring the correlation among successive generations. The covariance is related to the *exploration rate* of the algorithm, a scalar value between [0,1] and the only parameter of the algorithm that needs to be tuned. This version of CMA-ES does not support constrained optimization, which means that the optimized solutions that are not physically feasible on the real robot must be dropped and the learning algorithm restarted

3.6 Constrained CMA-ES

In robotics the notion of constraint arises in such diverse contexts and assuring feasibility plays a capital role when it comes to solve,

Assuring satisfaction constraints is a precondition that has to be met in order to transfer solution on the real robots. In our first work, [26], we employed CMA-ES to find a solution for the multi-tasks coordination problem. In this work we enforce constraints satisfaction through a constant penalties terms added to the fitness value of each candidates that violates one or more constraints. We observed that the proposed solution diminished the CMA-ES capabilities to find an optimal solution due to the introduction of artificial plateau in the fitness. So in the subsequent work we resolved to search for an approach that include constraint satisfaction directly in the exploration procedure.

In this paper, we adopt a different strategy and look explicitly for variants of CMA-ES that take into account the constraints in the exploration procedure. Our goals are: 1) to improve the efficiency of the optimization procedure exploiting the constraint information, and 2) to guarantee that every solution found by the stochastic optimization process lies in a region of the parameter space that satisfies all the constraints. Interestingly, we are not interested in algorithms that permit constraints relaxation (hence violation) to find a solution: this is typically the case of real-time quadratic solvers (*e.g.* quadprog and qpOASES).

Among the multitude of constrained black-box optimization algorithms, we focused on three variants of CMA-ES: a *vanilla penalty* CMA-ES, the CMA-ES with *adaptive penalty* approach proposed in [29] and the (1+1)-CMA-ES with *covariance constrained adaptation* proposed in [7]. The first is a baseline CMA-ES that applies a penalty to the fitness that is proportional to the constraint violation. The second method is similar in principle, but the penalty weights are changed following a heuristic that depends on the

CMA-ES with Vanilla Constraints

function CMA-ES-VC
  for each $gen = 1, \ldots, n_{GENERATIONS}$ **do**
    for $k = 1, \ldots, K$ **do**
      $\boldsymbol{\pi}_k \sim \mathcal{N}(\boldsymbol{\pi}, \varsigma^2 \boldsymbol{\Sigma})$
    end for
    for $k = 1, \ldots, K$ **do**
      $\phi_k = \phi(\boldsymbol{\pi}_k)$
      if ConstrViolation$(\boldsymbol{\pi}_k)$ **then**
        $\hat{J}_k =$ Penalty$(\boldsymbol{\pi}_k, J_k)$
      end if
    end for
    $\boldsymbol{\pi}_{1:K_e} =$ Sort$(\boldsymbol{\pi}_{k=1:K}, \hat{J}_{k=1:K})$
    $\boldsymbol{\pi}^{new} = \sum_{k=1}^{K_e} P_k \boldsymbol{\pi}_k$ with $\sum_{k=1}^{K_e} P_k = 1$
    $\boldsymbol{\Sigma}^{new} =$ UpdCov$(\boldsymbol{\pi}^{new}, P_{k=1:K_e})$
    $\varsigma^{new} =$ UpdStepSize(ς)
    $\bar{\boldsymbol{\pi}} = \bar{\boldsymbol{\pi}}^{new} \quad \boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{new} \quad \sigma = \sigma^{new}$
  end for
end function

FIGURE 3.6: Pseudocode of CMA-ES with Vanilla Constraints

constraint violation. The third does not rely on penalties but updates the covariance whenever a constraint is violated, to drive the exploration away from infeasible regions.

3.6.1 CMA-ES vanilla

The *vanilla penalty* functions method relies on adding a penalty term to the fitness of a candidate that depends on the constraints violation of the candidate. The method employs a penalized objective function $\hat{\phi}(\boldsymbol{\pi}_k) = \phi(\boldsymbol{\pi}_k) + l(\boldsymbol{\pi}_k)$ with the penalty factor $l(\boldsymbol{\pi}_k)$ defined as:

$$l(\boldsymbol{\pi}_k) = \sum_{i=1}^{n_{IC}} r_i \max(0, g_i(\boldsymbol{\pi}_k))^2 + \sum_{j=1}^{n_{EC}} c_j |h_j(\boldsymbol{\pi}_k)|$$

where r_i and c_i are positive constant values. In Fig. 4.5 we present a pseudo-code for this variant where we refer to the penalization routine with Penalty$(\cdot)$.

3.6.2 Adaptive CMA-ES

The previous method is by far the simplest and the most intuitive, as it applies a penalty that depends on the candidate $\boldsymbol{\pi}_k$. However, one may want to make the penalty term

CMA-ES with Adaptive Constraints

function CMA-ES-AC
  for each $gen = 1, \ldots, n_{GENERATIONS}$ **do**
    for $k = 1, \ldots, K$ **do**
      $\boldsymbol{\pi}_k \sim \mathcal{N}(\boldsymbol{\pi}, \varsigma^2 \boldsymbol{\Sigma})$
    end for
    for $k = 1, \ldots, K$ **do**
      $\phi_k = \phi(\boldsymbol{\pi}_k)$
    end for
    $[l, r, \overline{r}] = \textsc{CollectViolation}(\boldsymbol{\pi}_k, \boldsymbol{\epsilon})$
    $w^{new} = \textsc{UpdateWeight}(w, r, \overline{r})$
    $\hat{\phi}_k = \textsc{WeightPenalty}(w^{new}, l)$
    $\boldsymbol{\pi}_{1:K_e} = \textsc{Sort}(\boldsymbol{\pi}_{k=1:K}, \hat{J}_{k=1:K})$
    $\boldsymbol{\pi}^{new} = \sum_{k=1}^{K_e} P_k \boldsymbol{\pi}_k$ with $\sum_{k=1}^{K_e} P_k = 1$
    $\boldsymbol{\Sigma}^{new} = \textsc{UpdCov}(\boldsymbol{\pi}^{new}, P_{k=1:K_e})$
    $\varsigma^{new} = \textsc{UpdStepSize}(\varsigma)$
    $\bar{\boldsymbol{\pi}} = \bar{\boldsymbol{\pi}}^{new} \quad \boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{new} \quad \sigma = \sigma^{new}$
  end for
end function

FIGURE 3.7: Pseudocode for CMA-ES with Adaptive Constraints

variable, for example depending on the exploration path. Collange *et al.* [29] proposed a penalty function approach where a set of adaptive weights are tuned to prevent the search process from getting stuck in a local minima of the penalized fitness function $\hat{\phi}(\cdot)$. A penalized objective function $\hat{J}(\boldsymbol{\pi}_k)$ is therefore used. The key idea is that the penalty factor $l(\boldsymbol{\pi}_k)$ is built to consider the number of feasible solutions per each generation and the activation of each constraint, determined by a heuristic tuned by a user-defined ϵ_i. In particular, one assigns $l(\boldsymbol{\pi}_k) = \sum_{i=1}^{n_C} w_i [\gamma_i^+(\boldsymbol{\pi}_k)]^2$, where $w_i,\ i = 1, \ldots, n_C$ is the set of adaptive weights, and $\gamma_i^+(\cdot)$ is the positive part of the so-called *ϵ-normalized constraint values* γ_i, which are used to identify the active constraints. The ϵ-normalized constraint values γ_i are defined as:

$$\gamma_i = \begin{cases} [g_i(\boldsymbol{\pi}_k) + \epsilon_i]/\epsilon_i & \text{for inequality constraints} \\ |h_i(\boldsymbol{\pi}_k)|/\epsilon_i & \text{for equality constraints} \end{cases} . \tag{3.42}$$

The user can tune the definition of the constraint violations and the relaxation of the constraints through the parameters $\epsilon_i > 0, i = 1, \ldots, n_C$. For equality constraints $h_i(\cdot)$, a candidate solution π_k is labelled "feasible" if $0 \leq \gamma_i \leq 1$ and "infeasible" otherwise. For inequality constraints $g_i(\cdot)$, a candidate solution π_k is labelled "feasible" if $\gamma_i \leq 1$ and "infeasible" otherwise (Fig. 3.7).

The weights update is driven by the ratio of feasible solutions for all the constraints:

$$r_i^{feas} = \frac{\#\text{ feasible solutions for the } i\text{-th constraint in the curr. generation}}{K}$$
$$\bar{r}_i^{feas} = \text{average of } r_i^{feas} \text{ over the last } n_P + 2 \text{ generation}$$

If all the samples $\boldsymbol{\pi}_k$ with $k = 1, \ldots, K$ satisfy the i-th constraint, then $r_i^{feas} = 1$; otherwise $r_i^{feas} < 1$. So $\bar{r}_i^{feas} = 1$ only when all the samples satisfy the i-th constraint for $n_P + 2$ generations. Once this condition is met, the weight w_i related to the i-th constraint is decreased almost surely (in a statistical sense). The adaptation rule for the weight w_i after each generation is defined as: $w_i = w_i \ \exp(p_{target} - r_i^{feas})$, where p_{target} is a value that changes at each iteration according to $p_{target} = (1/Kn_P)^{1/\mathcal{D}}$, where $\mathcal{D}$ represents the cardinality of the elements' set that satisfies $i = 1, \ldots, n_C : \bar{r}_i^{feas} < 1$ and K is the number of samples. As a rule of thumb, when $r_i^{feas} > p_{target}$ the weight w_i decreases, otherwise it increases. In Fig. 4.5 we provide a pseudo-code for this variant, where we refer to the weight computation routine as UPDATEWEIGHT$(\cdot)$. For more detail on the method we refer to [29]. This method is more interesting since the penalty factor

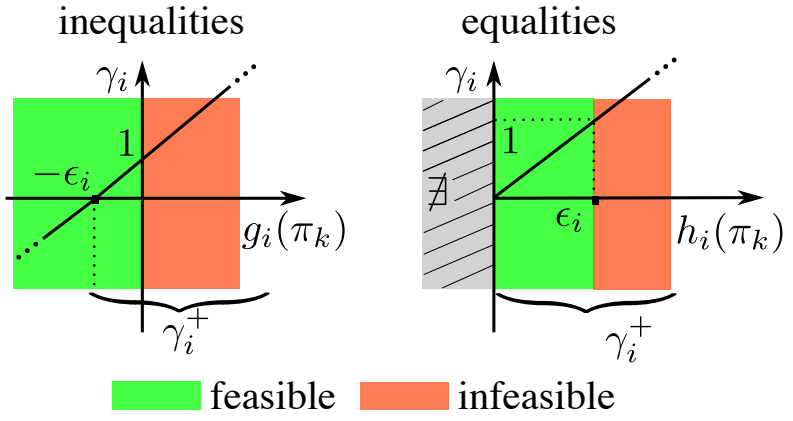


FIGURE 3.8: This illustration shows the relation between ϵ_i and γ_i for inequality and equality constraints as in Eq. 3.42. As described in Section 3.6.2, the green and red regions identify the constraint values that are respectively labeled as "feasible" and "infeasible". One may notice that ϵ_i induces a relaxation for the equality constraint: therefore it could be possible to label as feasible a solution that violates the constraint (how much depends on ϵ). On the contrary, it is noticeable that γ_i^+ in the inequality constraint also includes a boundary region determined by ϵ_i where the constraint is satisfied.

applied to the objective function changes during the optimization process depending on the number of feasible solutions that do not violate the constraints, considering the relaxation acting on the equality constraints. With respect to the vanilla method, the penalty factor here is not constant over the parameter space and depends on the exploration path in the parameter space. This decreases the possibility of getting stuck in local minima or flat areas of the fitness.

(1+1)-CMA-ES with Const. Cov. Adapt.

```
function (1+1)-CMA-ES-CCA
    π = FindFeasibleStartingPoint()
    for each gen = 1, ..., n_GENERATIONS do
        π_1 = π + ς Dz (Eq.3.43)
        if ConstrViolation(π_1) then
            D^new = UpCovConstr(); D = D^new
        else
            φ^new = φ(π_1)
            if φ^new > φ then
                D^new = UpCovSucc()
                ς^new = UpdStepSize(ς)
                π = π_1; D = D^new; ς = ς^new
            else if φ^new > φ^old then
                D^new = UpCovActive(); D = D^new
            end if
        end if
    end for
end function
```

FIGURE 3.9: Pseudocode for (1+1)-CMA-ES with Constrained Covariance Adaptation

3.6.3 (1+1)CMA-ES with Constrained Covariance Adaptation

The third method, proposed by Arnold *et al.* [7], is an extension of (1+1)-CMA-ES with active covariance adaptation [68]. As opposed to the other two methods, here we do not have a penalty factor, *i.e.*, the objective function is unchanged, but there is a different exploration strategy that exploits the constraints information to change the covariance and keep the optimization in a feasible region.

A notable difference with the classical CMA-ES is the fact that there is only one sample per generation ($\boldsymbol{\pi}_1$, therefore $K = 1$), that is generated according to the following rule:

$$\boldsymbol{\pi}_1 = \boldsymbol{\pi} + \varsigma \mathbf{D}\mathbf{z} \tag{3.43}$$

where $\mathbf{D}$ is the Cholesky factor of the covariance matrix $\boldsymbol{\Sigma} = \mathbf{D}^T\mathbf{D}$ and $\mathbf{z}$ is a standard normal distributed vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The algorithm stores the information about the successful steps in a *search path* $\mathbf{s} \in \mathbb{R}^{n_P}$. Each time a candidate outperforms the current best, $\mathbf{s}$ and $\mathbf{D}$ are updated (UpCovSucc in Fig. 4.5):

$$\begin{aligned}
\mathbf{s}^{new} &= (1-c)\mathbf{s}\sqrt{c(2-c)}\mathbf{D}\mathbf{z} \\
\mathbf{D}^{new} &= \sqrt{1-c^+_{cov}}\mathbf{D} + \frac{\sqrt{1-c^+_{cov}}}{||\mathbf{w}||^2}\left(\sqrt{1+\frac{c^+_{cov}||\mathbf{w}||^2}{1-c^+_{cov}}}-1\right)\mathbf{s}\mathbf{w}^T
\end{aligned}$$

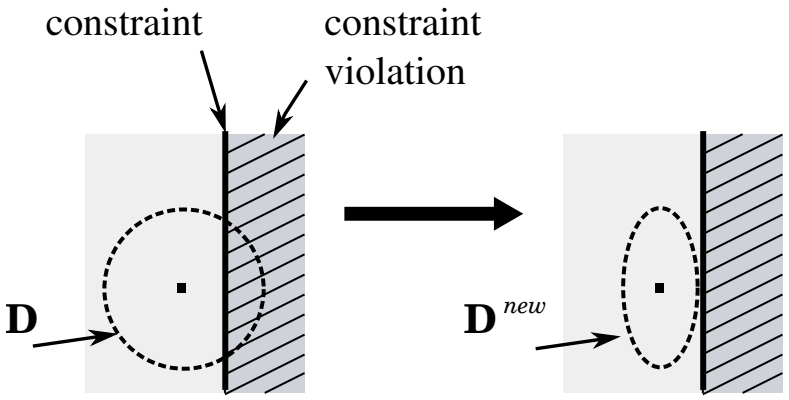


FIGURE 3.10: This illustration shows the effect of the covariance adaptation with constraints, as described in Section 3.6.3. A linear inequality constraint, represented by the vertical thick line, divides the parameter space into a region where the constraint is not violated (light grey) and a region where the constraint is violated (dark grey). The covariance $\mathbf{D}$ of the search distribution is updated in such a way that the successor samples will not fall into the region where the constraint is active: the updated covariance $\mathbf{D}^{new}$ is directed orthogonally with respect to the constraint.

where c^+_{cov} and c are both factors that control the update rate of $\mathbf{s}$ and $\mathbf{D}$ respectively, while $\mathbf{w} = \mathbf{D}^{-1}\mathbf{s}$. Instead, if the current candidate is feasible but its performance is lower than the predecessors, the Cholesky factor $\mathbf{D}$ is actively updated (UPCOVACTIVE in Fig. 4.5):

$$\mathbf{D}^{new} = \sqrt{1 + c^-_{cov}}\mathbf{D} + \frac{\sqrt{1 + c^-_{cov}}}{||\mathbf{z}||^2}\left(\sqrt{1 - \frac{c^-_{cov}||\mathbf{z}||^2}{1 + c^-_{cov}}} - 1\right)\mathbf{D}\mathbf{z}\mathbf{z}^T$$

where c^-_{cov} is again a constant that determines the update rate. In this case $\mathbf{s}$ is not updated because the current candidate is not better in terms of fitness.

To handle constraints, the key idea is to update the covariance matrix, by reducing the components of $\mathbf{Dz}$ in the direction that is orthogonal to the constraint whenever a constraint is violated, as illustrated in Fig. 3.10. Each time the j-th constraint is violated, we update the corresponding constraint vector $\mathbf{v_j} \in \mathbb{R}^{n_P}$ and the matrix $\mathbf{D}$ (UPCOVCONSTR in Fig. 4.5):

$$\mathbf{v}_j^{new} = (1 - c_c)\mathbf{v}_j + c_c\mathbf{D}\mathbf{z}$$
$$\mathbf{D}^{new} = \mathbf{D} - \frac{\beta}{\sum_{j=1}^{n_C} \mathbb{1}_{\{g_j(\boldsymbol{\pi}_1)>0\}}} \sum_{j=1}^{n_C} \mathbb{1}_{\{g_j(\boldsymbol{\pi}_1)>0\}} \frac{\mathbf{v}_j\mathbf{w}^T}{\mathbf{w}^T\mathbf{w}}$$

where c_c and β are constants that tune the update step respectively for $\mathbf{v}_j$ and $\mathbf{D}$, $\mathbf{w}_j = \mathbf{D}^{-1}\mathbf{v}_j$ and $\mathbb{1}_{\{g_j(\boldsymbol{\pi}_1)>0\}}$ is equal to one when $g_j(\boldsymbol{\pi}_1) > 0$ and zero otherwise.

In summary, the method searches for the optimal solution by testing one sample at the time and accounting for the constraints in the covariance adaptation to stay away

from infeasible regions. The algorithm is designed in such a way that the mean of the search distribution is updated only if the fitness improves and the candidate is a feasible solution; these two elements ensure that the solution of the optimization problem always satisfies the constraints. However, unlike the other methods, this requires a feasible[1] starting candidate to work, otherwise the exploration process quickly gets stuck. Hence, this method cannot be started from scratch or random values, but needs the pre-computation of a feasible starting point.

3.7 Constrained CMA-ES with memory

(1+1)CMA-ES with CCA has two known limitations: does not evolve if the starting point is unfeasible and it requires many samples to achieve local optimal solutions. In this manuscript, we propose an extension of (1+1)CMA-ES with CCA in order to mitigate these two issues. At the core of the new algorithm, there is a smart way to exploit the geometrical information that arises from the search distribution used to run a local meta-model optimization to boost the search for an optimal solution. This is obtained by combining (1+1)CMA-ES with CCA with Constrained Bayesian Optimization (CBO). To introduce CBO we need to introduce two basic concepts, Gaussian Process(GP) and Bayesian Optimization(BO). This introduction is based on [69].

3.7.1 Gaussian Process

GP is stochastic process that extends the multivariate Gaussian, and can be represented as an uncountable collection of random variables. Every subset of random variables in the GP displays a joint Gaussian distribution. GP as a generalization of Gaussian distribution can be viewed as a Gaussian prior over function and it is completely described by its mean and variance:

$$\tilde{\phi}(\mathbf{x}) = \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3.44}$$

where $\mathbf{x} \in \Pi \subseteq \mathbb{R}^{n_P}$ is an element of the parameter space, $m(\cdot)$ and $k(\cdot, \cdot)$ are respectively mean and variance of the stochastic process. Given a set of inputs $\mathbf{X} = [\mathbf{x_1}, ..., \mathbf{x_1}_d]$ and outputs $\boldsymbol{\phi}(\mathbf{X}) = [\phi(\mathbf{x_1}), ..., \phi(\mathbf{x_1}_d)]$ and one query point $\hat{\mathbf{x}}$ (of which the output is not known)the joint Gaussian distribution is defined as:

$$\begin{bmatrix} \boldsymbol{\phi}(\mathbf{X}) \\ \phi(\hat{\mathbf{x}}) \end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k}^T \\ \mathbf{k} & k(\hat{\mathbf{x}}, \hat{\mathbf{x}}) \end{bmatrix}\right) \tag{3.45}$$

[1] A candidate solution is feasible if it satisfies all the constraints.

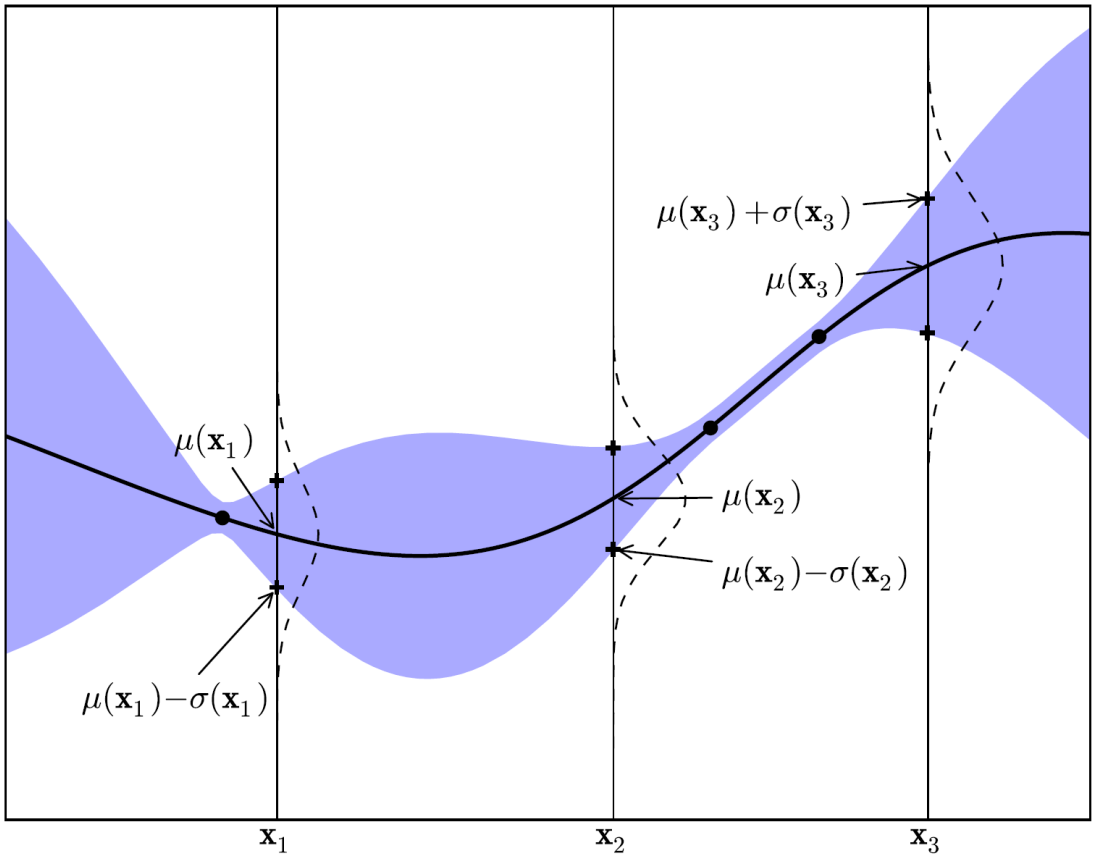


FIGURE 3.11: This illustration shows 1D Gaussian point with three query points. This picture is a visualization of the predictive. The thick black line represents the mean value predicted from the GP, and the blu area represents the one σ deviation from the mean values. At each query point $\mathbf{x}_{1:3}$ the univariate Gaussian that is computed from the predictive distribution is shown. This picture is taken from [69].

where:

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_d) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_d, \mathbf{x}_1) & \dots & k(\mathbf{x}_d, \mathbf{x}_d) \end{bmatrix} \tag{3.46}$$

$$\mathbf{k} = \begin{bmatrix} k(\hat{\mathbf{x}}, \mathbf{x}_1) & \dots & k(\hat{\mathbf{x}}, \mathbf{x}_d) \end{bmatrix} \tag{3.47}$$

Using the Sherman-Morrison-Woodbury formula [70] we can define the predictive distribution:

$$\mathcal{P}(\phi(\hat{\mathbf{x}})|\mathbf{X}, \hat{\mathbf{x}}) = \mathcal{N}(\mu(\hat{\mathbf{x}}), \sigma^2(\hat{\mathbf{x}})) \tag{3.48}$$

where:

$$\mu(\hat{\mathbf{x}}) = \mathbf{k}^T \mathbf{K}^{-1} f(\mathbf{X}) \tag{3.49}$$

$$\sigma^2(\hat{\mathbf{x}}) = k(\hat{\mathbf{x}}, \hat{\mathbf{x}}) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} \tag{3.50}$$

A full treatment of this topic is available at [70]

3.7.2 Bayesian Optimization

Bayesian optimization is a powerful tool to solve black box optimization problem. When we need to optimize an expensive objective function ϕ Bayesian Optimization turns out to be the method of choice to reduce the number of queries during the optimization. Bayesian Optimization employs a model of the objective function, usually a GP, and this model gives information across not yet evaluated points of the objective function. Based on this additional information a second acquisition function,$\tilde{\phi}$, is introduced to guide the process of selecting the next point given a utility criteria that has to be designed in advance. At each iteration, once the new point $\mathbf{x}^{new}$ is selected, the evaluation of this point is performed on the expensive function ϕ and the new input-output information is used to update the GP model. This process can be iterated until the problem's optimum is reached.

The performance of the entire process depends upon the choice of the acquisition function that enables active learning of the process. Let $\mathbf{x}^+$ be the current best point evaluated so far, $\hat{\mathbf{x}}$ a generic candidate point, we can introduce several acquisition functions well known from the literature.

3.7.2.1 Improvement Criteria

From Kushner [71] we introduce the *probability of improvement* over $\phi(\mathbf{x}^+)$ defined as:

$$PI(\hat{\mathbf{x}}) = \mathcal{P}(\tilde{\phi}(\hat{\mathbf{x}}) > \phi(\mathbf{x}^+)) \tag{3.51}$$

$$= \Phi\left(\frac{\mu(\hat{\mathbf{x}}) - \phi(\mathbf{x}^+)}{\sigma(\hat{\mathbf{x}})}\right) \tag{3.52}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ are respectively the mean and variance of the predictive distribution associated to the query point and $\Phi(\cdot)$ is the normal cumulative distribution function. The main issue about this approach is that this expression not allow for exploration. The PI criteria will privilege points with a an epsilon improvement over the current best while ignoring other points with a larger uncertainty but a smaller average value. To moderate this issue a trade-off parameter ξ is introduced:

$$PI(\hat{\mathbf{x}}) = \mathcal{P}(\tilde{\phi}(\hat{\mathbf{x}}) > \phi(\mathbf{x}^+ + \xi)) \tag{3.53}$$

$$= \Phi\left(\frac{\mu(\hat{\mathbf{x}}) - \phi(\mathbf{x}^+) - \xi}{\sigma(\hat{\mathbf{x}})}\right) \tag{3.54}$$

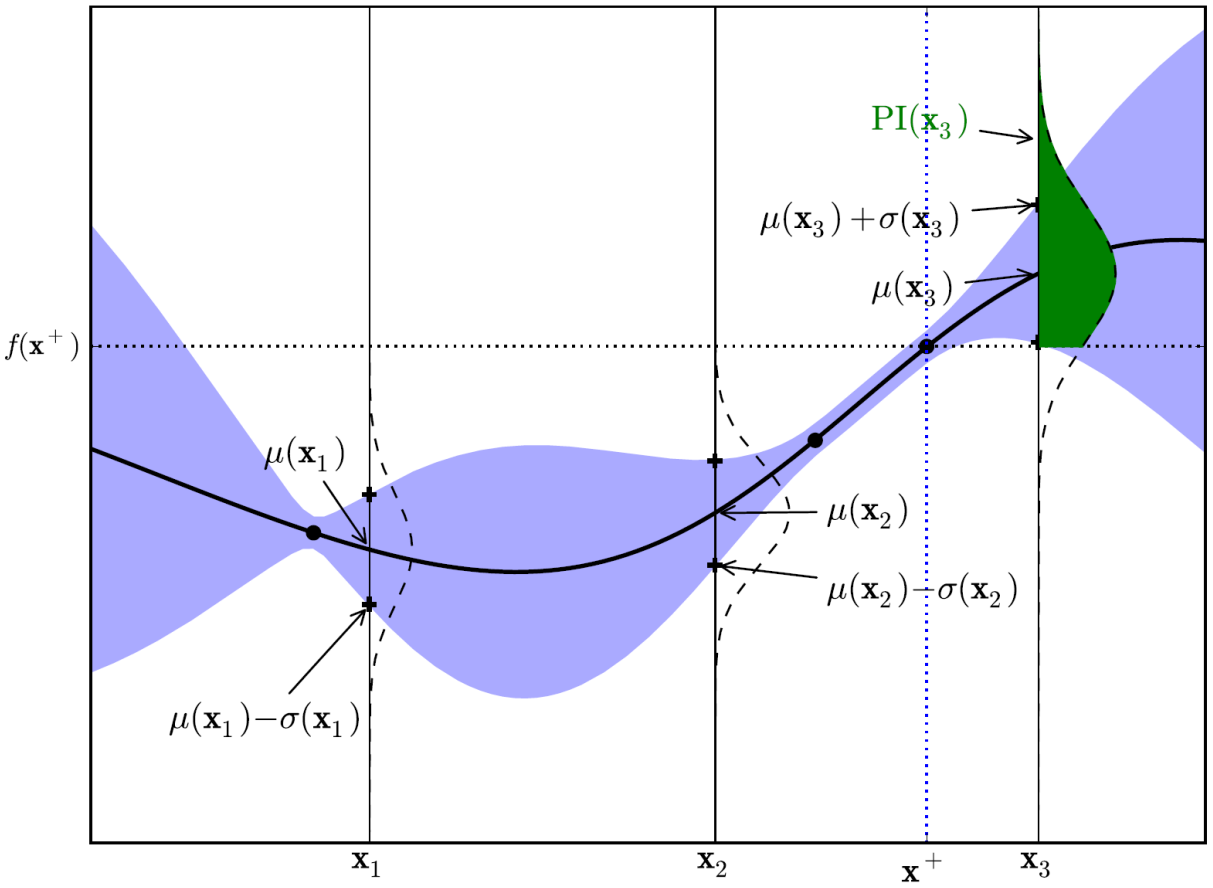


FIGURE 3.12: This illustration shows the same 1D gaussian process from Figure 3.11. Here it is shown how the PI acquisition function is computed. The current best candidate is $\mathbf{x}^+$. The dark green area under the dashed gaussian distribution represents a measure of improvement. As you can see from the picture the PI is mostly an exploitation method because the mean is the main driver for the selection of the next point. For this reason it is necessary to introduce a parameter ξ to promote an exploration behaviour. This image is taken from [69].

. The choice of ξ is left to the user. Kushner suggests use a scheduling scheme for the value of ξ that slowly reduces over time. A better choice for the acquisition is one that tries to minimize the expected deviation from the true optimal solution $\mathbf{x^o}$. This selection criteria will be:

$$\mathbf{x}^{new} = \underset{\hat{\mathbf{x}}}{\operatorname{argmin}}[\mathbb{E}(\|\hat{\phi}(\hat{\mathbf{x}}) - \phi(\mathbf{x}^o)\|)] \tag{3.55}$$

$$= \underset{\hat{\mathbf{x}}}{\operatorname{argmin}} \int \|\tilde{\phi}(\hat{\mathbf{x}}) - \phi(\mathbf{x}^o)\| \mathcal{P}(\tilde{\phi}(\hat{\mathbf{x}})|\mathbf{X}, \hat{\mathbf{x}}) d\tilde{\phi}(\hat{\mathbf{x}}) \tag{3.56}$$

To avoid a myopic approach we should apply a recursive approach over many samples into the future. Due to the computational cost of this approach Mockus [72] introduces the *Expected Improvement* criteria. The improvement $\mathcal{I}$ of the candidate point $\hat{\mathbf{x}}$ is defined as

$$\mathcal{I} = \max\left(0, \phi(\mathbf{x}^+) - \tilde{\phi}(\hat{\mathbf{x}})\right) \tag{3.57}$$

and the new query point is determined by computing the Expected Improvement

$$EI(\hat{\mathbf{x}}) = \mathbb{E}(\mathcal{I}(\hat{\mathbf{x}})|\hat{\mathbf{x}}) \tag{3.58}$$

In the work of Mockus [72] and Jones [73] a closed form of the Expected Improvement is given:

$$EI(\hat{\mathbf{x}}) = \sigma(\hat{\mathbf{x}})Z\Phi(Z) + \phi(Z) \tag{3.59}$$

$$Z = \frac{\mu(\hat{\mathbf{x}}) - \phi(\mathbf{x}^+)}{\sigma(\hat{\mathbf{x}})} \tag{3.60}$$

where $\phi(\cdot)$ denotes the pdf of the normal distribution. In the same way of Probability of Improvement Lizotte [74] introduces an extension of EI that gives the user a control capability over the exploitation-exploration trade-off. Lizzote introduces a parameter ξ:

$$EI(\hat{\mathbf{x}}) = \sigma(\hat{\mathbf{x}})Z\Phi(Z) + \phi(Z) \tag{3.61}$$

$$Z = \frac{\mu(\hat{\mathbf{x}}) - \phi(\mathbf{x}^+) - \xi}{\sigma(\hat{\mathbf{x}})} \tag{3.62}$$

3.7.2.2 Confidence Bound Criteria

Another common criteria for an acquisition function is the *confidence bound*. For a minimization problem we introduce the *lower confidence bound*:

$$LCB(\hat{\mathbf{x}}) = \mu(\hat{\mathbf{x}}) - \kappa\sigma(\hat{\mathbf{x}}) \tag{3.63}$$

where $\kappa \geq 0$. If we are interested in a maximization, a *upper confidence bound* is introduced

$$LCB(\hat{\mathbf{x}}) = \mu(\hat{\mathbf{x}}) + \kappa\sigma(\hat{\mathbf{x}}) \tag{3.64}$$

In both cases κ is a parameter left to the user's choice. From Srinivas *et al.* [75] if Bayesian Optimization problem is recasted in a multi-armed bandit setting we can redefine the acquisition function as instantaneous regret:

$$r(\hat{\mathbf{x}}) = \phi(\mathbf{x}^o) - \tilde{\phi}(\hat{\mathbf{x}}) \tag{3.65}$$

Given this formulation, the goal is to find

$$\min \sum_{t}^{T} r(\hat{\mathbf{x}}_\mathbf{t}) \tag{3.66}$$

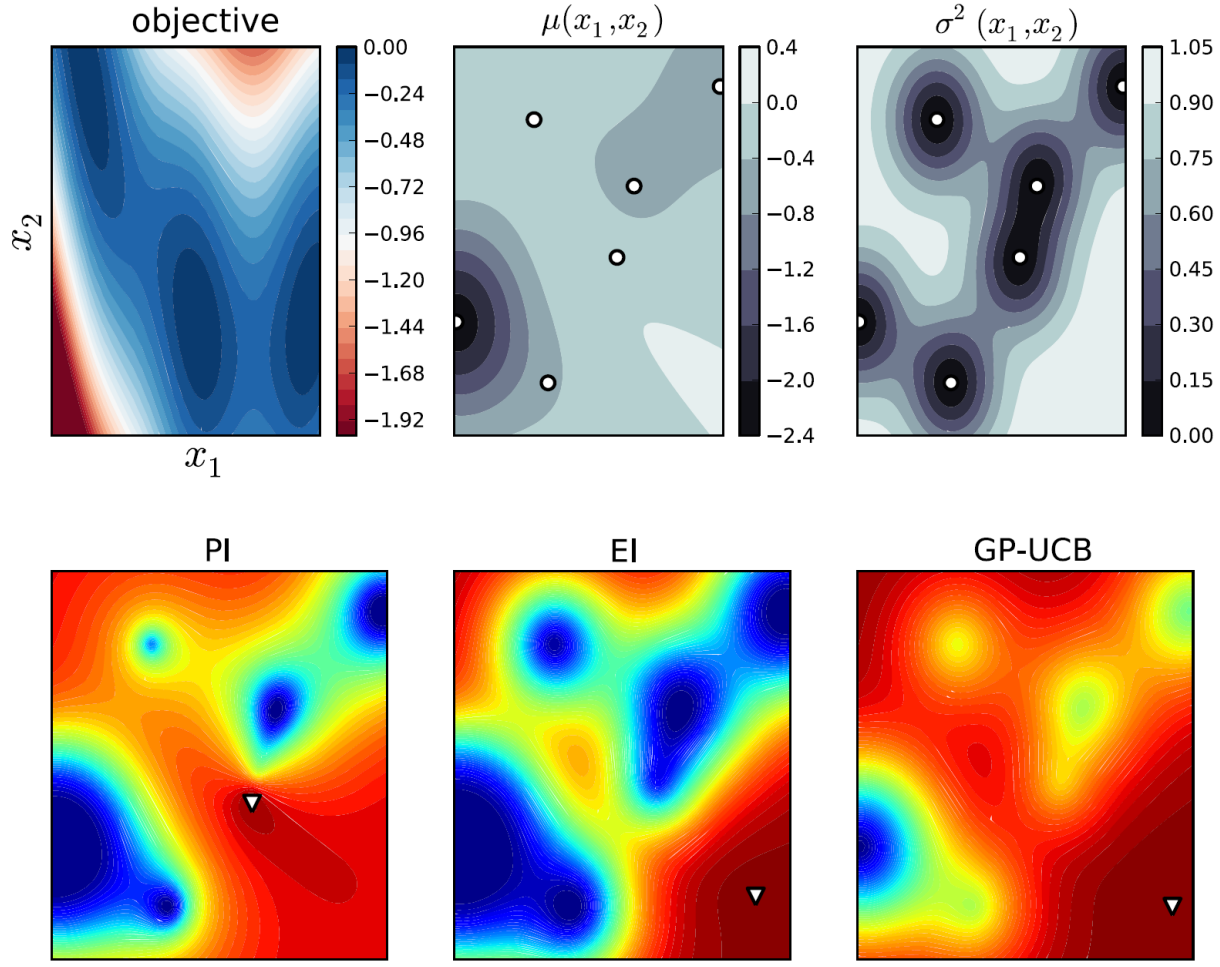


FIGURE 3.13: This is an example of three different acquisition functions for a 2D optimization problem. The first image in the first row represents the value of the objective (ground truth) and the second and the third ones show respectively the mean and the variance of the GP surrogate model. In the bottom rows the heatmap in each picture shows where the most desirable points to pick next are. The best value for the current iteration is represented through a white triangle. The represented acquisition functions are Probabilty of Improvement (PI), Expected Improvement (EI) and Upper Confidence Bound (GP-UCB). This picture is taken from [69].

Where T is the total number of "trials" that are performed on the process. The *Upper Confidence Bound* selection criteria introduced by Srinivas *et al.*

$$GP - UCB = \mu(\hat{\mathbf{x}}) + \sqrt{\nu\tau_t}\sigma(\hat{\mathbf{x}}) \tag{3.67}$$

with $\kappa_t = \sqrt{\nu\tau_t}$ and ν an hyperparameter greater than 0. It is possible to prove with high probability that with certain choices for ν and τ_t ($\nu = 1$ and $\tau_t = 2$) the selection criteria displays no regret ($\lim_{T\to\text{inf}} R_t/T = 0$) where R_t

$$R_t = \sum_{t=1}^{T} r(\hat{\mathbf{x}}_t) \tag{3.68}$$

3.7.3 Constrained Bayesian Optimization

Here we introduce the method proposed by Gardner *et al.* [76]. In this work the authors introduce a novel acquisition function designed for expensive constrained black-box optimization. The novel acquisition criteria extends the EI approach in order to take into account the problem's constraints for the selection of the next query point. For one constraint, $g(\hat{\mathbf{x}})$, they introduce the *constrained improvement* for the $\hat{\mathbf{x}}$ candidate:

$$\mathcal{I}_c = \mathbb{1}_{\{g(\hat{\mathbf{x}})\leq 0\}}(\hat{\mathbf{x}}) \max\left[0, \phi(\mathbf{x}^+) - \phi(\hat{\mathbf{x}})\right] \tag{3.69}$$

$$= \mathbb{1}_{\{g(\hat{\mathbf{x}})\leq 0\}}(\hat{\mathbf{x}})\mathcal{I}(\hat{\mathbf{x}}) \tag{3.70}$$

where $\mathbf{x}^+$ represents the current best *feasible* candidate and $\mathbb{1}_{\{g(\hat{\mathbf{x}})\leq 0\}}$ is an indicator function that is one when the candidate is feasible otherwise is zero. We provide a model both for the objective functions $\tilde{\phi}(\cdot)$ and the constraints $\tilde{g}(\cdot)$ using a set of GPs. Every iteration for each new candidate x^{new}, we evaluate both the objective function and of all the constraints and we update the set of input-output value $\mathbf{X}_\phi$ and $\mathbf{X}_g$ respectively for the predictive distribution of the objective function $\mathcal{P}(\phi(\hat{\mathbf{x}})|\mathbf{X}_\phi, \hat{\mathbf{x}})$ and the predictive distribution of the constraints $\mathcal{P}(g(\hat{\mathbf{x}})|\mathbf{X}_g, \hat{\mathbf{x}})$. Given this model the improvement becomes

$$\tilde{\mathcal{I}}_c = \tilde{\mathbb{1}}_{\{g(\hat{\mathbf{x}})\leq 0\}} \max\left[0, \phi(\mathbf{x}^+) - \tilde{\phi}(\hat{\mathbf{x}})\right] \tag{3.71}$$

$$= \tilde{\mathbb{1}}_{\{g(\hat{\mathbf{x}})\leq 0\}}\tilde{\mathcal{I}} \tag{3.72}$$

where $\tilde{\mathbb{1}}_{\{g(\hat{\mathbf{x}})\leq 0\}}$ is defined as a Bernoulli random variable

$$PF = \mathcal{P}(\tilde{g}(\hat{\mathbf{x}}) \leq 0) = \int_{-\inf}^{0} \mathcal{P}(g(\hat{\mathbf{x}})|\mathbf{X}_g, \hat{\mathbf{x}}) d\tilde{g}(\hat{\mathbf{x}}) \tag{3.73}$$

Because of the marginal Gaussianity of the $\tilde{g}_i$ constraints model, PF is a univariate Gaussian distribution. The *Expected Constrained Improvement* is derived as:

$$ECI(\hat{\mathbf{x}}) = \mathbb{E}\left[\tilde{\mathcal{I}}_c|(\hat{\mathbf{x}})\right] \tag{3.74}$$

$$= \mathbb{E}\left[\tilde{\mathbb{1}}_{\{g(\hat{\mathbf{x}})\leq 0\}}\tilde{\mathcal{I}}|(\hat{\mathbf{x}})\right] \tag{3.75}$$

$$= \mathbb{E}\left[\tilde{\mathbb{1}}_{\{g(\hat{\mathbf{x}})\leq 0\}}|(\hat{\mathbf{x}})\right] \mathbb{E}\left[\tilde{\mathcal{I}}|(\hat{\mathbf{x}})\right] \tag{3.76}$$

$$= PF(\hat{\mathbf{x}})EI(\hat{\mathbf{x}}) \tag{3.77}$$

where the third line comes from the fact that objective function and constraints are conditional independent. Therefore the ECI corresponds to the expected improvements over the feasible current best $\mathbf{x}^+$ weighted by the probability that the candidate point is

feasible. Even if the infeasible points are not a desirable outcome during the optimization process they prove to be useful in determining the shape of the constraint functions. This property renders the algorithm independent from the starting point because it is not necessary to sample feasible region to to discover them.

For multiple constraints $\mathbf{g} = \mathbf{0}$ where $\mathbf{g} = [g1, \ldots, g_{n_{IC}}]$ and $\mathbf{0} \in \mathbb{R}^{n_{IC}}$, the expectation of the Bernoulli random variable is redefined as

$$PF = \mathbb{E}\left[\tilde{\mathbb{1}}_{\{\mathbf{g}(\hat{\mathbf{x}}) \leq \mathbf{0}\}} | \hat{\mathbf{x}}\right] = \mathcal{P}(\tilde{g}_1(\hat{\mathbf{x}}) \leq 0, \ldots, \tilde{g}_{n_{IC}}(\hat{\mathbf{x}}) \leq 0) \tag{3.78}$$

$$= \prod_{i=1}^{n_{IC}} \mathcal{P}(\tilde{g}_i(\hat{\mathbf{x}}) \leq 0) \tag{3.79}$$

where the probability factorization is made possible because the constraints are conditionally independent over $\hat{\mathbf{x}}$. Each probability that composes the product is a univariate Gaussian distribution.

3.7.4 The algorithm

Two main issues negatively affect the overall perfomances of (1+1)CMA-ES with CCA. (1+1)CMA-ES with CCA displays a very slow converge rate and it does not work if the starting point does not satisfy all the constraints of the problem. In this section we propose an algorithm to solve black box constrained optimization that by combining the Constrained Bayesian Optimization framework with (1+1)CMA-ES with CCA, aims to find a solution faster than the other CMA-ES approaches with a random starting point. This algorithm is designed to search for many local optimal solutions in order to provide a richer description of the objective function landscape. Here we introduce the concept of Particle $\mathcal{PA}_i$: each $\mathcal{PA}_i$ represents an instance of a (1+1)CMA-ES with CCA and λ is a user defined parameter that controls the maximum number of particles that can be deployed during one optimization run. We designed a set of particle's *death* and *birth* rules that were set in place to avoid wasting computational time over non informative particles and to not incur in early convergence states. Because our algorithm employs a CBO scheme, a set of surrogate function modeled as Gaussian Process are introduced. We define a model for the objective function $\tilde{\phi}$ and for the constraints $\tilde{g}_i$ with $i = 1, \ldots, n_{IC}$. At the end of each iteration we update all the models adding to the table of input-output values (respectively $\mathbf{X}_\phi$ and $\mathbf{X}_{g_i}$ for objective function and constraints) the last tested point with the associated objective function value. In the algorithm we identify 4 different phases and a set of transitions that take place only if certain conditions are met. The 4 stages are:

- initialization

(1+1)BO-CMA-ES with CCA

function (1+1)BO-CMA-ES
  deploy,bootstrapping = true
  btIt = 0
  $\#ofparticles = 0$
  CreateGPs()
  for each $gen = 1, \ldots, n_{GENERATIONS}$ **do**
    if $btIt > btTresh$ && $bootstrapping$ **then**
      state = bootstrapping
      act = BtActionSelector()
      $curPartIt = 0$
      $\boldsymbol{\pi}_1$ == DoAction($state, act, curPartIt$)
      $\phi^{new}, \phi^{new}_{constr}, violation$ = Execution($\boldsymbol{\pi_1}$)
      if !$violation$ **then**
        bootstrapping = false
      end if
    else
      if deploy && (#ofparticles < $lambda$) **then**
        state = deploy
        act = DeployActionSelector()
        $curPartIt = 0$
        $\boldsymbol{\pi}_1$ == DoAction($state, act, curPartIt$)
        $\phi^{new}, \phi^{new}_{constr}, violation$ = Execution($\boldsymbol{\pi_1}$)
        $btIt = btIt + 1$
        if $feasiblecandidate$ **then**
          AddNewParticle($\boldsymbol{\pi}_1$)
          #ofparticles = #ofparticles + 1
        end if
      else
        deploy = false;
        state = optimization
        [$act, curPartIt$] = OptimActionSelector()
        $\boldsymbol{\pi}_1$ == DoAction($state, act, curPartIt$)
        $\phi^{new}, \phi^{new}_{constr}, violation$ = Execution($\boldsymbol{\pi_1}$)
        if !$violation$ && $act == global$ **then**
          if $\phi^{new} > \phi^{best}$ **then**
            AddNewParticle($\boldsymbol{\pi}_1$)
          end if
        end if
        if (#ofparticles > 1) **then**
          PruneParticles()
        end if
      end if
    end if
    if ($act == particleSample \parallel act == Local$) **then**
      UpdateParticle($state, act, curPartIt$)
    end if
    UpdateGPs($\phi^{new}, \phi^{new}_{constr}, violation$)
  end for
end function

Figure 3.14: Pseudo code for (1+1)BO-CMA-ES

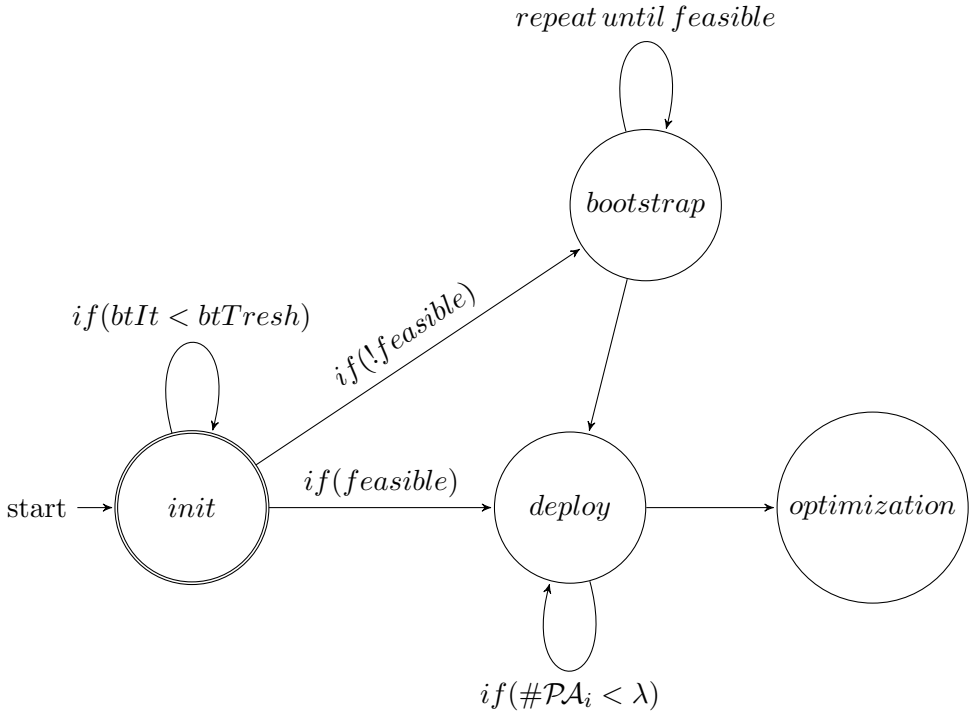


FIGURE 3.15: This figure shows the states that compose our algorithm and the conditions that have to be met in order to evolve from one state to the next

- bootstrapping
- particles deploy
- constrained optimization

During the initialization phase, the algorithm looks for a feasible solution. If it fails, we enter in the bootstrapping mode where we cast an unconstrained optimization problem specifically designed for the research of a feasible solution. Once a feasible solution is found (during the initialization or after the bootstrapping phase) we start to deploy a λ number of $\mathcal{PA}_i$ inside the feasible region. Once we reach the desired number of particles, during the optimization phase we make the particles evolve until certain conditions are met or we attain the maximum number of generations. Two user defined parameters control the transitions among the phases:

- λ determines how many particles $\mathcal{PA}_i$ have to be deployed before entering in the constrained optimization phase. In this algorithm we introduce a set of user defined parameters to control the state machine evolution over generations.
- $btTresh$ controls how many generations we can wait, during the initialization phase, before starting the bootstrapping phase if we do not find a feasible starting point.

In this algorithm we distinguish between local and global search. We refer to the former when the search for the optimal point is restricted to the neighbourhood of the current particle, while we consider the latter when the search region coincides with the entire hyperrectangle of the parameters space. We designed only 1 action that perform a global search and we conceive 2 actions that implements a local search search. A Bayesian optimization (constrained or unconstrained), performed on the entire parameters space, represents our global action. It can differs among all the phases for the kind of acquisition function that is employed for the search. The first local action is performed every time we sample from the search distribution associated to each particle to enact one step of (1+1)CMA-ES. The second local action is represented as a Bayesian Optimization (constrained or unconstrained) performed in a neighbourhood of the particle that we are considering. given the Search Distribution (SD) of the Particle $\mathcal{PA}_i$

$$SD = \mathcal{N}(\boldsymbol{\pi}_i, \varsigma_i^2 \boldsymbol{\Sigma}_i) \tag{3.80}$$

Where π_i is the mean, $\boldsymbol{\Sigma}_i$ is its covariance matrix and ς_i is a multiplication factor that is used inside CMA-ES to set the step size value. Here we can define a local reference frame, *loc*, centred in the mean of the particle and oriented as the principal direction of the covariance matrix. Therefore we compute the Eigen Decomposition of the convariance matrix $\boldsymbol{\Sigma}_i$

$$\boldsymbol{\Sigma}_i = \mathbf{V}_i \text{diag}(\boldsymbol{\delta}_i) \mathbf{V}_i^{-1} \tag{3.81}$$

where $\mathbf{V}_i$ is the eigenvectors matrix while $\boldsymbol{\delta}_i$ is the vector of eigenvalues. With this information we can compute the new local bounding box built around the hyper ellipsoid of the gaussian search distribution associated to the i-th particle. The local hyperrectangle bounds are defined as

$$\mathbf{b}_i^{max} = \text{diag}(|\boldsymbol{\delta}_i|) \varsigma_i k_m k_u \tag{3.82}$$

$$\mathbf{b}_i^{min} = -\text{diag}(|\boldsymbol{\delta}_i|) \varsigma_i k_m k_u \tag{3.83}$$

$$\tag{3.84}$$

where k_u is a user defined parameter that is used to inflate or deflate the local bounding box, finally k_m represents a particular Mahalanobis distance that describes an ellipsoid centred at the mean of the distribution. In order to compute the k_m we have to define a specific confidence interval that determines the the probability mass enclosed inside the ellipsoid. For our purpose we use a confidence interval equal to 0.95. k_m is defined as

$$k_m = \sqrt{\text{invChi}(0.95, n_p)}; \tag{3.85}$$

where n_p is dimension of the multivariate distribution and invChi is the inverse cumulative distribution function of the chi squared distribution up to the confidence value because the Mahalanobis distance of a multivariate gaussian is distributed according to the Chi-squared distribution with n_p degrees of freedom. When we perform the BO in the local bounding box To speed up the optimization of the acquisition function we compute at runtime for both the objective function and the constraints a local Gaussian process. For the i-th particle the new GPs are defined by considering only a subset of points of $\mathbf{X}_{(\cdot)}$ lying in a neighbourhood of the search distribution

$$\mathbf{X}_{(\cdot)}^{loc} \subseteq \mathbf{X}_{(\cdot)} : ||\mathbf{x} - \boldsymbol{\pi}_i|| \leq \max\left(\mathbf{b}_i^{max}\right) \quad \text{with } x \in \mathbf{X}_{(\cdot)} \tag{3.86}$$

where $\max\left(\mathbf{b}_i^{max}\right)$ represents an hypersphere centred in the mean of the SD with radius equal to the biggest axis of the local bounding box. To guarantee that the optimal solution will belong to the local hyperrectangle we perform the optimization of the acquisition function directly in the local reference frame *loc*. For that reason, given a generic acquisition function built on the restricted Gaussian Process $ac(\hat{\mathbf{x}}, \mathbf{X}^{loc})$ and the set of point that belongs to the local hyperrectangle in the local reference frame $\mathbf{b}_i^{min} \leq \hat{\mathbf{x}}^{loc} \leq \mathbf{b}_i^{max}$ we define a local acquisition function $ac_l(\hat{\mathbf{x}}^{loc}, \mathbf{X}^{loc})$

$$ac_l(\hat{\mathbf{x}}^{loc}, \mathbf{X}^{loc}) = ac\left[t(\hat{\mathbf{x}}^{loc}), \mathbf{X}^{loc}\right] \tag{3.87}$$

where $t(\cdot)$ represents the rototraslation from the local reference to the original frame

$$t(\hat{\mathbf{x}}^{loc}) = \boldsymbol{\pi}_i + \mathbf{V}_i \hat{\mathbf{x}}^{loc} \tag{3.88}$$

where the matrix of the eigenvector $\mathbf{V}_i$ acts as rotation matrix. By interleaving random sampling (according to the search distribution) and "informed" sampling (through the Bayesian optimization framework) we mitigate the slow converge rate issue that afflicts every CMA-ES approach. We implemented different heuristics for each phase to switch among actions.

Hereafter we introduce, with a detailed description, each phase that compose our method.

3.7.4.1 Initialization Phase

As we stated before, each particle need a feasible starting point to properly work. So we need to find λ feasible points to start the optimization phase. For our problem we make the hypothesis that the problem is well posed so at least one feasible region exists. In order to do so we search for a feasible optimal solution using an instance of Constrained Bayesian optimization 3.7.3. Here we use a different acquisition function from the one

used in [76]. To push the exploratory behaviour of the search, we introduce another constrained acquisition function called Constrained Variance (CV) defined as:

$$CV = \mathbb{E}\left[\tilde{\mathbb{1}}_{\{\mathbf{g}(\hat{\mathbf{x}})\leq\mathbf{0}\}}|\hat{\mathbf{x}}\right]\sigma_\phi(\hat{\mathbf{x}}) \tag{3.89}$$

$$= \prod_{i=1}^{n_{IC}} \mathcal{P}(\tilde{g}_i(\hat{\mathbf{x}}) \leq 0)\sigma_\phi(\hat{\mathbf{x}}) \tag{3.90}$$

where $\sigma_\phi(\hat{\mathbf{x}})$ is the variance of the predictive distribution of the objective function's Gaussian Process model. Weighting the variance of the surrogate objective function with the probability of constraints satisfaction leads to a behaviour that try to look into the regions with less points with a preference for the feasible ones. If we find at least one or more feasible regions before reaching the $btTresh$ maximum number of generations, we reach the deploy phase. Otherwise if, the free region is too narrow or the number of maximum allowed iterations is too small we start the bootstrapping phase.

3.7.4.2 Bootstrapping Phase

If we fail to find a feasible starting point the algorithm is provided with a specific procedure to manage this occurrence. In this section we apply the issue we employ a simple *divide et impera* idea to render the issue easier to manage. During thi phase we recast a new unconstrained optimization problem where the sum of constraints violations is the new fitness function:

$$\phi_{bs} = \begin{cases} -\sum_t^T \sum_{i=1}^{n_C} |\hat{e}(t,i,\boldsymbol{\pi})|, & \text{if constr. viol.} \\ -\sum_t^T \sum_{i=1}^{n_C} |\hat{s}(t,i,\boldsymbol{\pi})|, & \text{if no constr. viol.} \end{cases}$$

where T is the number of control steps, $\hat{e}(t,i,\boldsymbol{\pi}) = \mathbb{1}_{\{g_i(\boldsymbol{\pi})>0\}}\, g_i(\boldsymbol{\pi})$ and $\hat{e}(t,i,\boldsymbol{\pi}) = \mathbb{1}_{\{h_i(\boldsymbol{\pi})\neq 0\}}\, h_i(\boldsymbol{\pi})$ are respectively the inequality and equality constraints that are not satisfied at time t and $\hat{s}(t,i,\boldsymbol{\pi}) = \mathbb{1}_{\{g_i(\boldsymbol{\pi})<=0\}}\, g_i(\boldsymbol{\pi})$, $\hat{s}(t,i,\boldsymbol{\pi}) = \mathbb{1}_{\{h_i(\boldsymbol{\pi})=0\}}\, h_i(\boldsymbol{\pi})$ represent all the satisfied constraints at time t. Even though during the Initialization phase we employed only one global action here To solve this unconstrained optimization we employ all the actions that are at our disposal. In this phase we start the optimization process by randomly selecting a starting point. From this point we initialize only one particle $\mathcal{PA}_{bt}$ that is an instance of unconstrained (1+1)CMA-ES. After a fixed amount of generations (in our experiment we use a value of 10 max iterations) where we let the (1+1)CMA-ES evolve. when we reach the maximum number of iteration we perform a global BO if the objective function value associated to the current search distribution mean is lower than a certain threshold. Otherwise, we use a local BO in the neighbourhood of the

search distribution. After one of the two BO actions is performed we set back to the local (1+1)CMA-ES evolution. For both the local BO and the global one we use as an acquisition function the Probability of Constraint Satisfaction (PCS) defined as

$$PCS = \prod_{i=1}^{n_{IC}} \mathcal{P}(\tilde{g}_i(\hat{\mathbf{x}}) \leq 0) \tag{3.91}$$

3.7.4.3 Deploy Phase

Once a feasible solution is found (directly after the Initialization Phase or because a Bootstrapping become necessary) the algorithm starts to add particles until we reach the maximum number requested, λ. In this phase the Heuristic for the action selection is slightly different than if the previous phase was the initialization or the Bootstrapping phase. If the algorithm moved directly from the initialization phase we keep looking for other feasible point using the CBO with the acquisition function already employed during the Initialization phase (see Eq. 3.89). Although if the algorithm requested a bootstrapping it means that the feasible region is narrow in respect to the global bounding box that contains the parameter space. So in this case a better strategy consists in looking for other starting points around the bootstrapping particle. As such, we alternate between one evolution of the $\mathcal{PA}_{bt}$ and on local CBO around the bootstrapping particle. Every time a feasible point is sampled we add a new particle. For the local CBO we introduce a new acquisition function called Maximum Constrained Distance (MCD)

$$MCD = \prod_{i=1}^{n_{IC}} \mathcal{P}(\tilde{g}_i(\hat{\mathbf{x}}) \leq 0) \sum_{i=1}^{n_d} ||\pi_i - \hat{\mathbf{x}}|| \tag{3.92}$$

where $n_d < \lambda$ is the total number of particles already deployed. In this way the research for the next candidate during the local CBO is skewed to the points that are feasible and have the biggest distance from the particle already deployed. In this way we try to obtain a good coverage of the feasible region.

3.7.4.4 Optimization phase

After collecting λ particles, the algorithm proceeds with the optimization of the original constrained problem. In advance, the optimal value of the objective function is unknown so we need to iterate through every action during the entire optimization process. We start by applying a simple round-robin heuristics to evolve every particle in our set. When each particle has evolved a fixed amount of times we alternate between a round of local CBO for each particle and a global CBO search. For both local and global CBO, we employ the ECI acquisition function from Gardner *etal*. [76] (3.7.3). For the

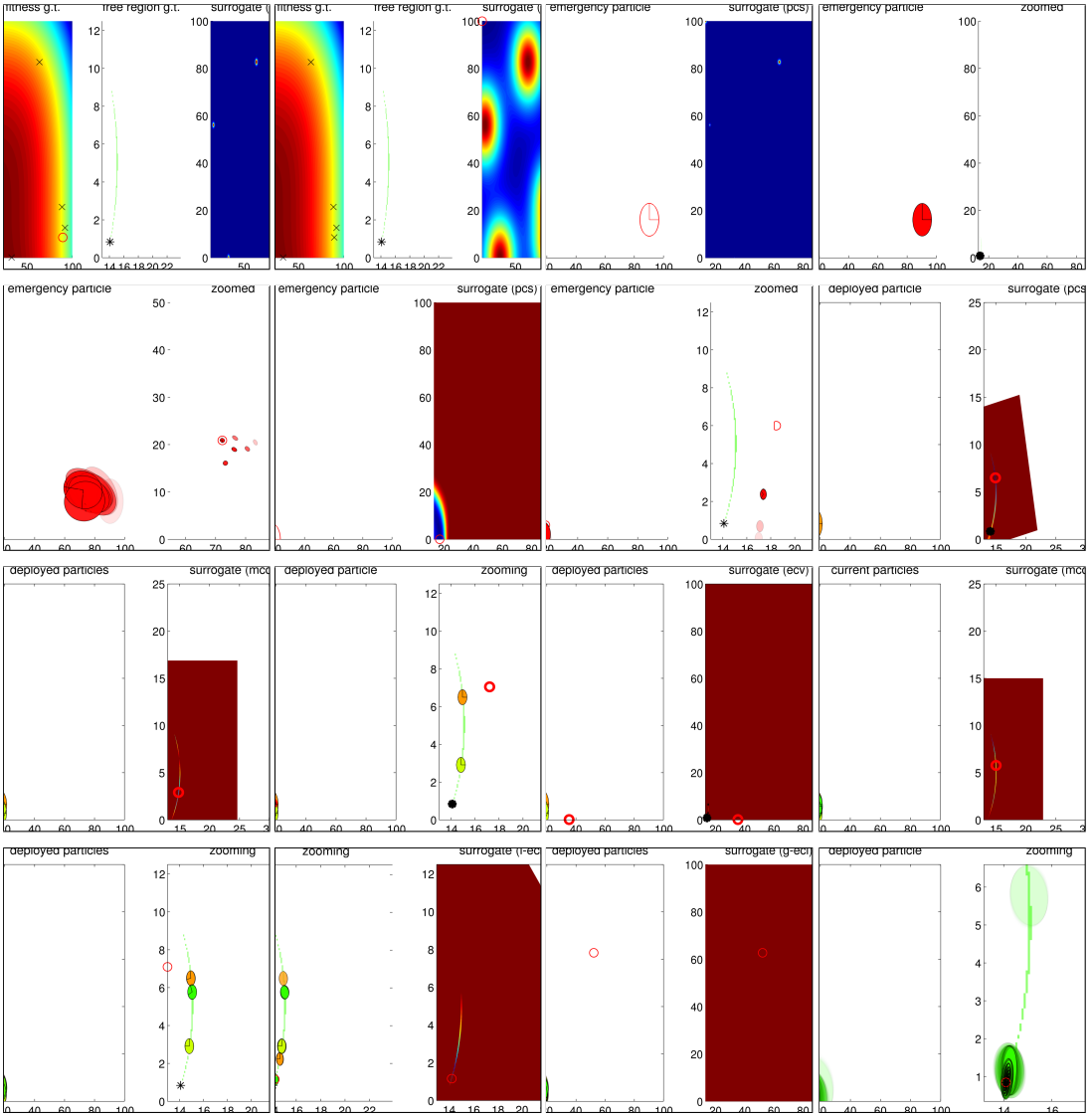


FIGURE 3.16: in this figure we show the different phases of the proposed algorithm. In the first row we start by using a global constrained BO to look for a solution that satisfies all the constraints. Because it fails, we enter in bootstrapping mode where only one particle is deployed to look for a free solution. In this phase we interleave after 10 sampling from the search distribution a global and a local BO (in this order). Finally, once a solution is found, in the third and fourth rows the algorithm performs the deployment of three particle to look for an optimal solution of the constrained optimal problem that is eventually found by the green particle (in the last figure at the bottom right)

global CBO we consider as current best sample $\mathbf{x}^+$ the highest fitness value among all particles. In order to reduce the execution cost when we have to deal with numerous particles we introduce a set of death rules. The first rule aims to reduce the particle redundancy. If the distance between to particles become lower than a certain threshold we remove the particle with the lower fitness value. The second rule is designed to remove the particle that reach a local optima. For each particle we record the consecutive number of generations where the particle did not evolve. If this number becomes greater

than a predefined maximum value we drop the current particles. If the number of active particles drops to 1 we do not apply any more any death rules and we execute the algorithm until the maximum number of generation is reached. To allow for late discovery of new feasible region, we defined a birth rule. Each time that the algorithm employs a global CBO action if the sampled point is feasible we add a new particle if and only if we have less than λ active particles.

The proposed algorithm presents little criticality. The great number of user defined parameters introduced in the proposed algorithm nullify one of the best feature of CMA-ES (and all its variations), having few parameters to tune. Even if some of the new parameters introduced are not critical, others can play an essential role in order to find an optimal solution. To overcome this, we plan to introduce a multi armed bandit approach for the action selection in each phase (similar to [77]). The computational time is another issue that affects the proposed method. This is due to the introduction of the BO framework. When the training set starts to grow in size the it takes more time to compute the prediction distribution. To reduce this problem we can use C++ implementation of BO that presents has better performance than the Matlab implementation.

3.8 Benchmarking the Algorithms

In this section we test the algorithms described in Section 3.6 to decide which one better suits our problem. We compare their performances on five different benchmarks:

- g07: $n_P = 10, n_{IC} = 8, n_{EC} = 0$
- g09: $n_P = 7, n_{IC} = 4, n_{EC} = 0$
- HB: $n_P = 5, n_{IC} = 6, n_{EC} = 3$
- RB1: $n_P = 15, n_{IC} = 32, n_{EC} = 0$
- RB2: $n_P = 15, n_{IC} = 50, n_{EC} = 0$

The first three are classical benchmarks for constrained optimization [7], that is analytic problems with known optimal solutions; the last two are new benchmarks that we designed *ad hoc* to evaluate the performance of the algorithms on robotic problems with growing complexity. RB1 is a problem inspired by our previous work [26] where a KUKA LWR (7DOF) has to reach a goal position with its end-effector behind an obstacle, while satisfying constraints of joint position limits, joint torque limits and obstacle avoidance. RB2 has a similar setting with the addition of a second obstacle to avoid and another set of constraints coming from joint velocity limits. To compare the performance of the algorithms on these benchmarks, we define the following metrics:

- m_1: *distance from the optimal solution*, defined as $m_1 = \|\boldsymbol{\pi}^\circ - \boldsymbol{\pi}^*\|$, where $\boldsymbol{\pi}^\circ$ is the optimal solution (known) and $\boldsymbol{\pi}^*$ the best solution found by the constrained optimization algorithm;

- m_2: *constraint violations*, defined as $m_2 = \sum_{i=1}^{nC} |\hat{e}(i, \boldsymbol{\pi}^*)|$, where $\hat{e}(i, \boldsymbol{\pi}) = \mathbf{1}_{g_i(\boldsymbol{\pi})>0}\, g_i(\boldsymbol{\pi})$ for the inequality constraints and $\hat{e}(i, \boldsymbol{\pi}) = \mathbf{1}_{h_i(\boldsymbol{\pi})\neq 0}\, h_i(\boldsymbol{\pi})$ for the equality constraints — basically it sums all the constraints that are violated;

- m_3: *number of steps to converge*, or *settling time*, defined as $m_3 = n_{sc}(\delta)$, the number of steps after which the fitness function reaches a steady state condition, *i.e.*, its value is bounded between $\pm\delta\%$ of the steady state value — here, we set $\delta = 2.5$;

- m_4: *best fitness*, defined as $m_4 = J(\pi^*)$, *i.e.*, the fitness of the best solution found by the constrained optimization algorithm.

3.8.1 Benchmarking (1+1)CMA-ES with CCA

In this section we provide some benchmarking results for (1+1)CMA-ES with CCA. Here we compare (1+1)CMA-ES with CCA with to other variants of CMA-ES for constrained black box optimization: CMA-ES-AC and CMA-ES-VC. To provide a baseline, we use the (deterministic) constrained optimization function *fmincon* in Matlab, using the SQP method. This is a suitable choice because it does not require the gradient of the objective function for non-linear constrained optimization problem with nonlinear constraints.

Since (1+1)-CMA-ES with covariance constrained adaptation (Section 3.6.3) needs a feasible candidate solution as a starting point, in order to make a fair comparison all the algorithms start from the same initial position. We perform 40 repetitions of the optimization process per each test problem for each algorithm with an exploration rate of 0.1 and a 5000 samples to assure the convergence of the methods.

Fig. 3.17 shows the results of the numerical experiments with the five benchmarks. The top row reports on the results for g07, g09 and HB with metrics m_1, m_2, m_3, while the bottom row reports on the results for the robotics benchmarks RB1 and RB2, with metrics m_2, m_3, m_4 (m_1 cannot be used in this case because the optimal solution $\boldsymbol{\pi}^\circ$ is not known). We also compared the four algorithms in terms of computational time, and did not find significant differences (for example, the optimal solution for RB2 is found on average in ≈1.7e+04 s for the CMA-ES variants and 1.9e+04 s for *fmincon* on a i5 laptop with Matlab).

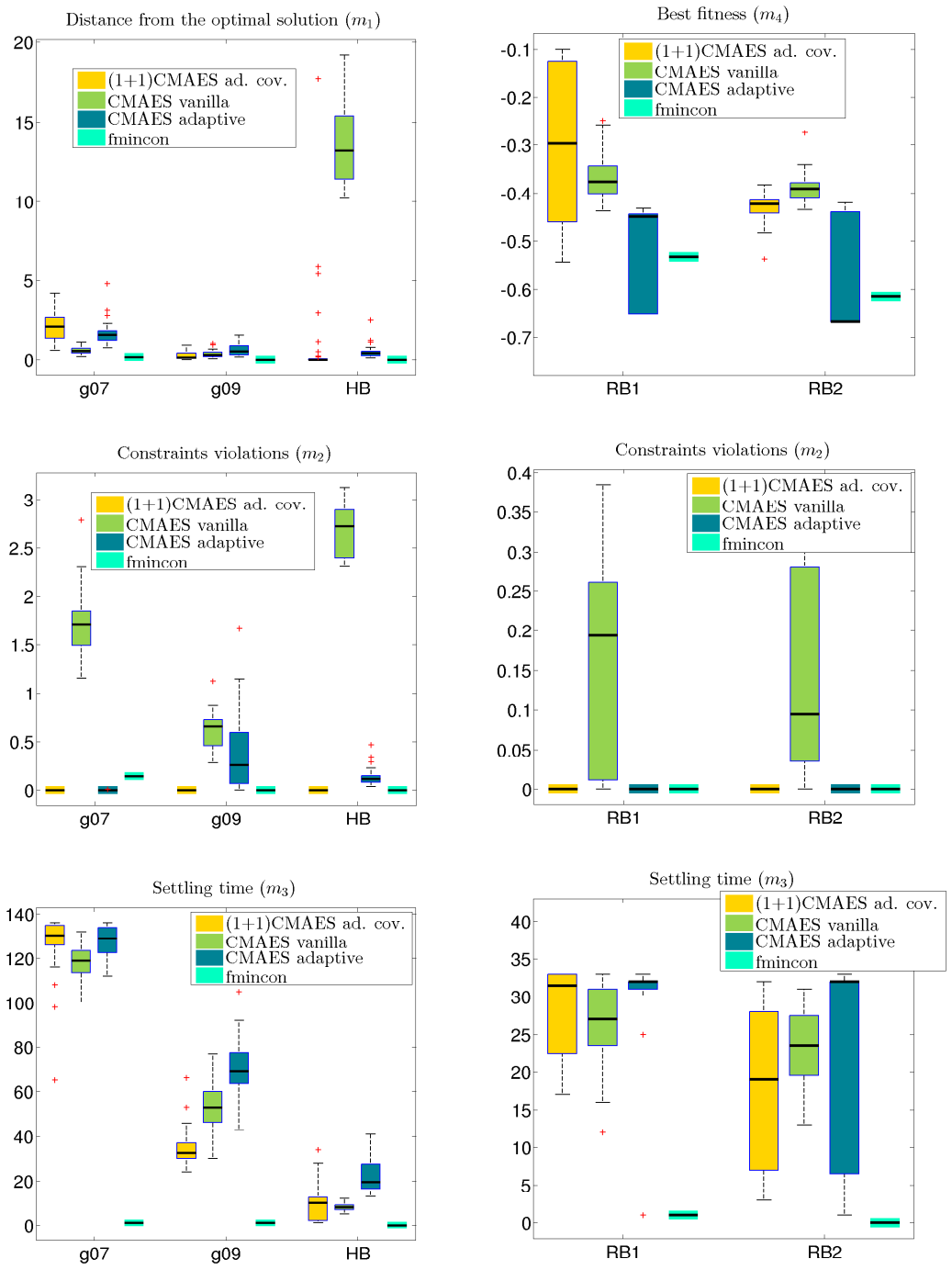


FIGURE 3.17: Performance comparison of the three constrained CMA-ES algorithms and the baseline *fmincon* algorithm from Matlab using the SQD method. The top row reports on the results on three standard analytical constrained optimization benchmarks (g07, g09, HB - see [7]). The bottom row reports on the results on two robotics benchmarks (RB1, RB2) that we designed *ad hoc* to evaluate the performance of the algorithms on robotics problems.

(1+1)-CMA-ES with covariance constrained adaptation offers the best trade-off between

performance and constraints' satisfaction both on the analytic and the robotic benchmarks. It always ensures full satisfaction of the constraints while the other methods sometimes fail. Its settling time is comparable to the other stochastic algorithms, while *fmincon* converges faster. *fmincon* could seem more appealing, but on the robotic benchmarks its best fitness is lower and actually quite close to the fitness of the starting point (meaning that the algorithm does not really "explore"). Therefore *fmincon* does not seem a suitable candidate for solving robotic problems with a lot of constraints.

The different performances of the algorithms in the analytic and robotic benchmarks confirm the benefit gained by designing two new robotics benchmarks RB1,RB2. Overall, considering the zero constraint violations and the capability of finding a good solution, we choose to use (1+1)-CMA-ES with covariance constrained adaptation for our experiments with the iCub robot.

3.8.2 Benchmarking (1+1)BO-CMA-ES

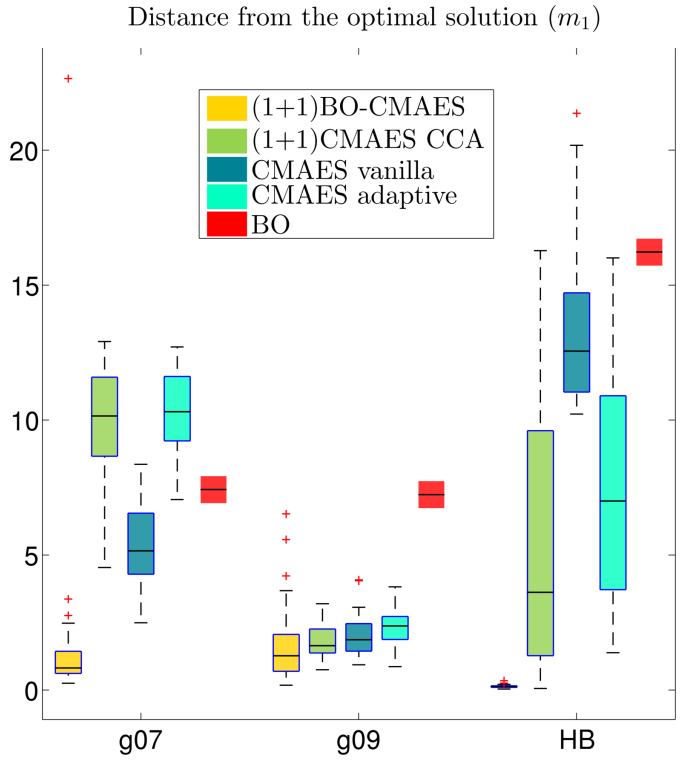


FIGURE 3.18: This illustration shows the result for the m_1 metrics. Our method outperforms all the others in term of distance from the optimal solutions. Here the smaller the value the better.

In this section we provide benchmarking results for the updated version of (1+1)CMA-ES with CCA. In this benchmark we present some preliminary results on a set of analytical constrained optimization problems and we do not provide any comparison on the

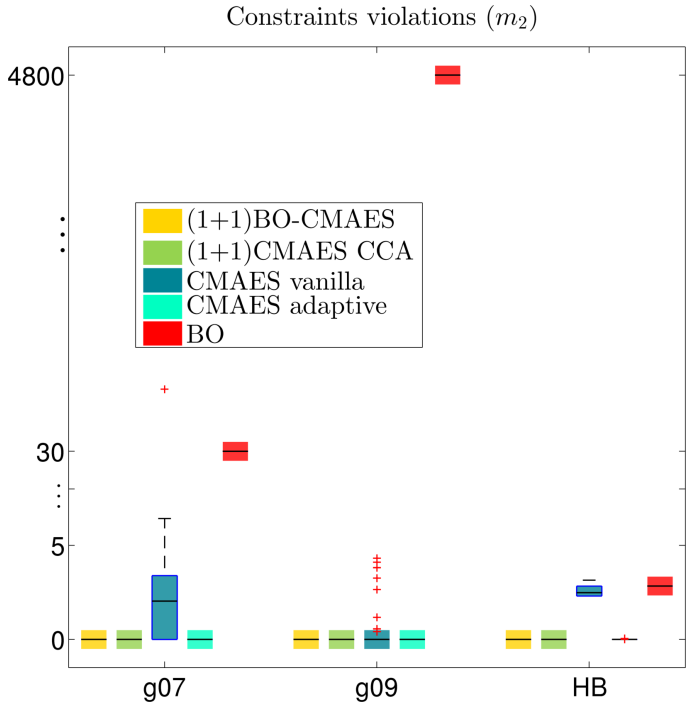


FIGURE 3.19: In this figure the metric m_2 results are presented. While all the CMA-ES methods satisfy the constraints (with the exception of CMA-ES-VC), the CBO approach greatly violates them. This happens because for the CBO method there are no guarantees that such occurrences may not happen.

robotics benchmarks. Here we compare (1+1)BO-CMA-ES with CCA and other variants of CMA-ES for constrained black box optimization: CMA-ES-AC, CMA-ES-VC, (1+1)CMA-ES with CCA and a plain implementation of Constrained Bayesian Optimization with an ECI acquisition function. We perform 45 repetitions for each benchmark problem with an exploration rate of 0.1 and 300 maximum allowed iterations for each algorithm. Because (1+1)CMA-ES with CCA need a feasible starting solution, we used the same starting point for every algorithm except for (1+1)BO-CMA-ES and the CBO algorithm. For CBO and (1+1)BO-CMA-ES we provide only 5 points with the associated fitness to initialize the Gaussian Processes for the objective function and all the constraints. For the CBO method we perform only one experiment because we use the same starting points. In this case CBO is strictly deterministic. In this benchmark we used only 3 particles for (1+1)BO-CMA-ES because we know in andvance that this analytical problems present only one optimum. Even if the comparison was not completely fair, the results show that, given the small iteration budget, the proposed method outperforms the others in terms of optimality of the solutions found. From the Figure 3.18 it is clear that even for the first analytical problem (where the feasible region is in a narrow portion of the search space compared to the entire bounding box) our method show to get extremely close to the optimal solution while the others method do not present the same capability. For the other two benchmark functions, g09*and*HB, our

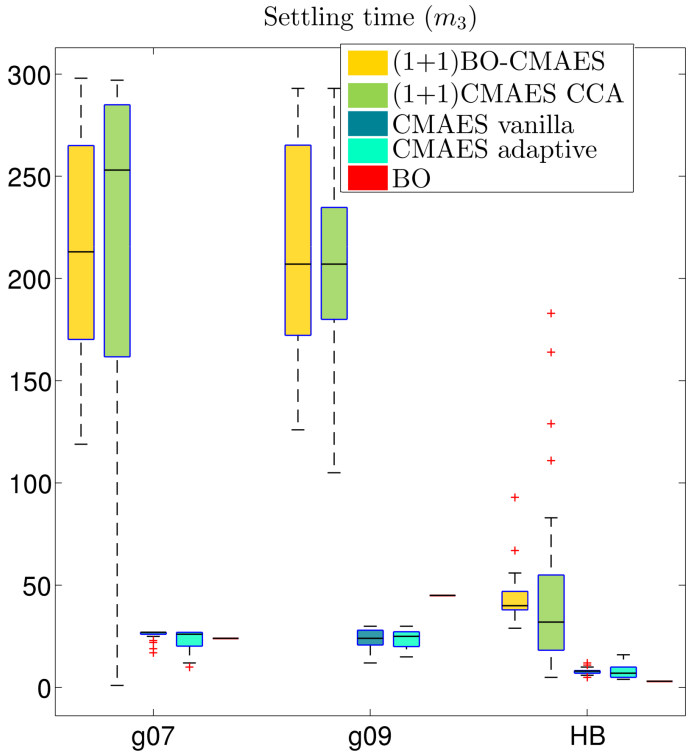


FIGURE 3.20: This image presents the results for the convergence rate metric m_3. Even if our method shows a worse performance than the other approaches, for the converge rate computation we take into account all the iterations spent during the bootstrapping phase. On the contrary the other algorithms started directly from a feasible solution that reduces the overall iteration cost.

algorithm reach better results especially for the last function where we got the exact optimal solution. This behaviour is possible because in our algorithm we leverage on local and global Bayesian optimization moves that reduce the number of required sample. For the metrics m_2 we get comparable results among the CMA-ES based algorithms while the CBO alone results in huge constraints violations. For the metric m_3, our algorithm shows a similar converge of (1+1)CMA-ES with CCA with a difference: by starting from an infeasible solution we spend some iteration for the bootstrapping procedure and at the end we reach a better optimal solution. If we consider the computational time to reach a solution, our algorithm is much slower than the others (except for the CBO). This problem is related to the computational cost associated to the calculation of the predictive distribution for the acquisition function optimization inside CBO. We believe that we can mitigate this issue by migrating our code from Matlab to better performing language like C++ or Java.

Chapter 4

Experiments

4.1 Optimizing Soft Task Priorities Without Constraints

In this section we discuss our experiments on learning the task priorities. We start showing on a simulated Kinova Jaco arm that our learning method improves the performance of the movement in terms of fitness values, over existing task priorities that have been manually tuned. We also show that the optimized trajectories are robust with respect to the initialization of the learning process. We compare on a real Jaco arm some typical learned policies with the manually tuned one, showing that our method improves the real robot motion. Finally, we compare on a simulated Kuka LWR the performance of our method with the state-of-art GHC controller [19] . We show that our method is not only better in terms of performance, but also computationally 10 times faster.

4.1.1 Learning the Task Priorities for the Kinova Jaco Arm

The setting for the first experiment is shown in Figure 4.1. The Kinova Jaco arm (6 DoF), starting from its zero configuration, must reach a desired position behind a wall with its end-effector. The goal position is difficult to reach, and the robot kinematics is such that it is not straightforward to manually design a trajectory that does not collide with the obstacle and brings the hand to the goal.

There are 3 given elementary tasks. The first is about reaching the Cartesian pose $\mathbf{p}^*$ =[0, -0.63, 0.7] with the end-effector (*goal*). The second is about reaching the Cartesian pose [-0.31, -0.47, 0.58] with the 4th link. The third is about keeping the joint configuration [120, 116, 90, 0, 0, 0] (degrees).

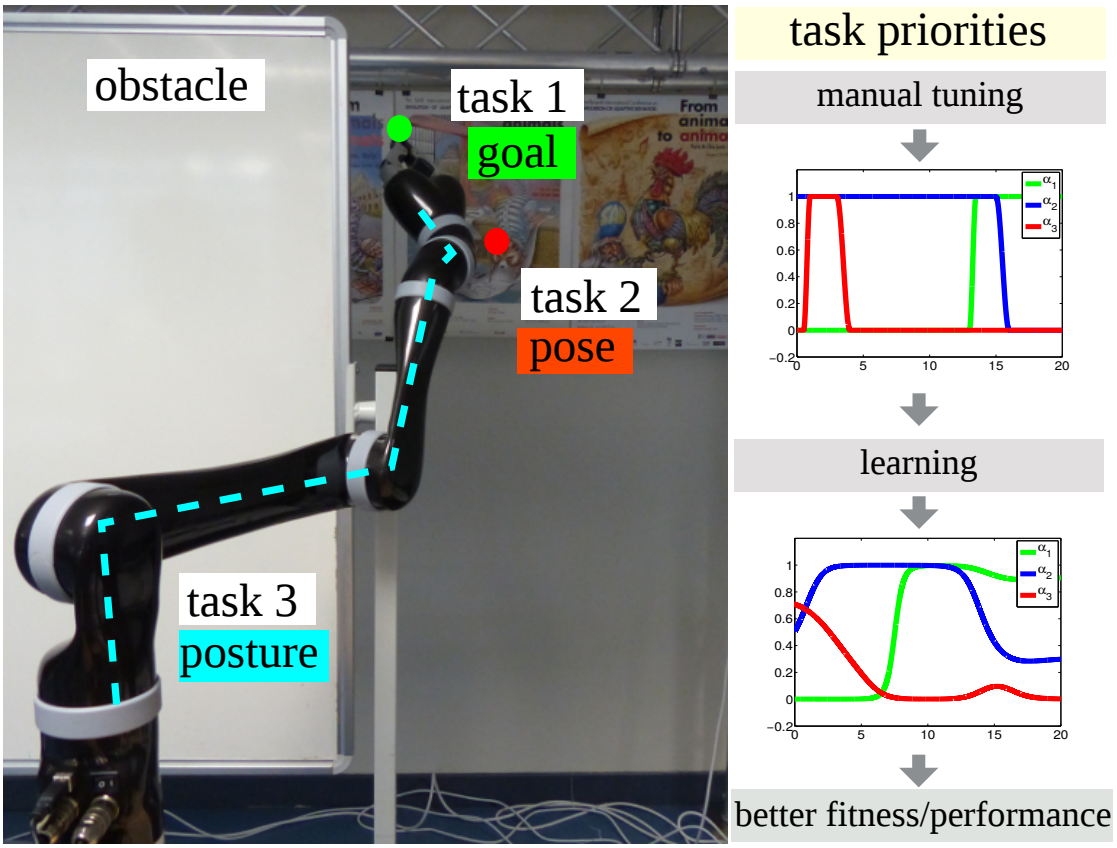


FIGURE 4.1: The Jaco arm must reach a goal behind a wall (obstacle) while fulfilling a pose task on joint 4 and a full posture task. The initial sequencing of task priorities is not efficient. Our method allows the automatic learning of the temporal profiles of the task priorities from scratch.

We design the following fitness function $\phi \in [-1, 0]$:

$$\phi(\mathbf{q}_{1,\dots,T}, \mathbf{u}_{1,\dots,T}) = -\frac{1}{2}\left(\frac{\sum_i^T \|\mathbf{p}_i - \mathbf{p}^*\|}{\epsilon_{\max}} + \frac{\sum_i^T \|\mathbf{u}_i\|_2^2}{u_{\max}}\right)$$

where T is the number of control steps (the task duration is 20 seconds, and we control at 10ms), $\mathbf{p}_i$ describes the end-effector position at time i and $\mathbf{p}^*$ is the goal position, $\|\cdot\|_2^2$ is the square of the ℓ^2 norm and $\epsilon_{\max}^{-1}$ and $u_{\max}^{-1}$ are two scaling factors. The first term of ϕ penalizes the cumulated distance from the goal, while the second term penalizes the global control effort. To ensure that the generated controls are feasible for the real Jaco robot, we set the fitness to -1 whenever the generated policy violates one of the robot constraints: a collision with the environment, joints position ranges and maximum joint torques. This ad-hoc solution is also a consequence of the learning algorithm. Fig. 4.2 shows the average fitness value computed over the eleven optimized trajectories satisfying the constraints, found on 100 restarts of CMA-ES.

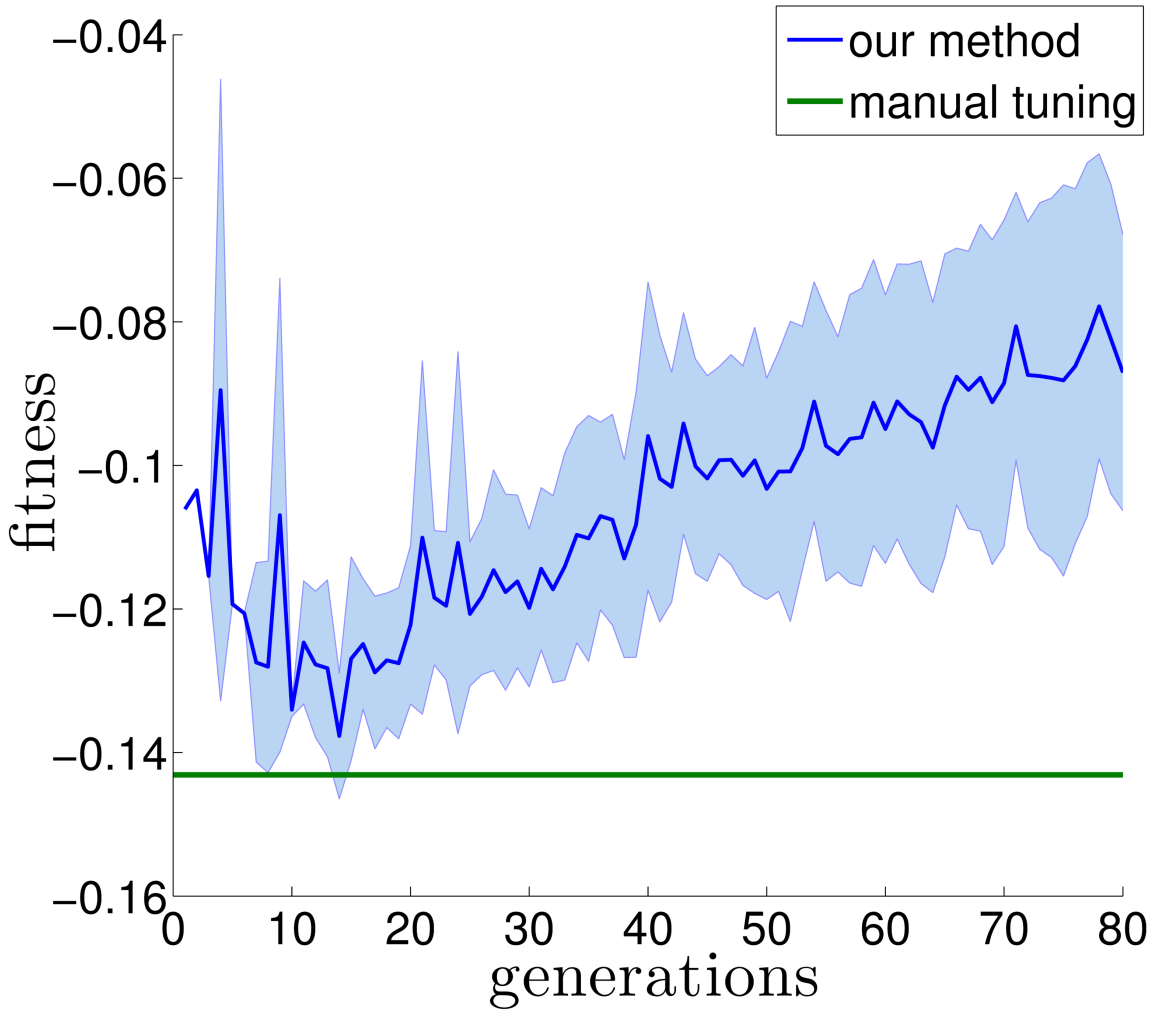


FIGURE 4.2: Average fitness value for the task priorities learned with our method, for the 3-tasks experiment with the simulated Jaco arm. The horizontal line indicates the fitness of the manually tuned solution. The mean and standard deviation of the fitness for the learned policies is computed over 100 restarts of CMA-ES, each with 80 generations and random initialization of the parameters. We only retained the fitness for the experiments that provided solutions satisfying the real robot constraints.

4.1.2 Robustness of the Learning Process

Different profiles of the task priorities can yield similar movements of the robot. It is however important to show that the learning process is able to optimize the task priorities in a robust way, that is providing similar optimal solutions. We therefore execute N=100 replicates of the experiment in Section 4.1.1, with a simulated Jaco arm and three tasks. In each experiment, CMA-ES runs for 100 generations with an exploration rate of 0.5. The parameters are randomly initialized. We compute the average and standard deviation of the solutions that satisfy the robot and task constraints. Figure 4.3 shows the average end-effector trajectory in the Cartesian space and the corresponding joint

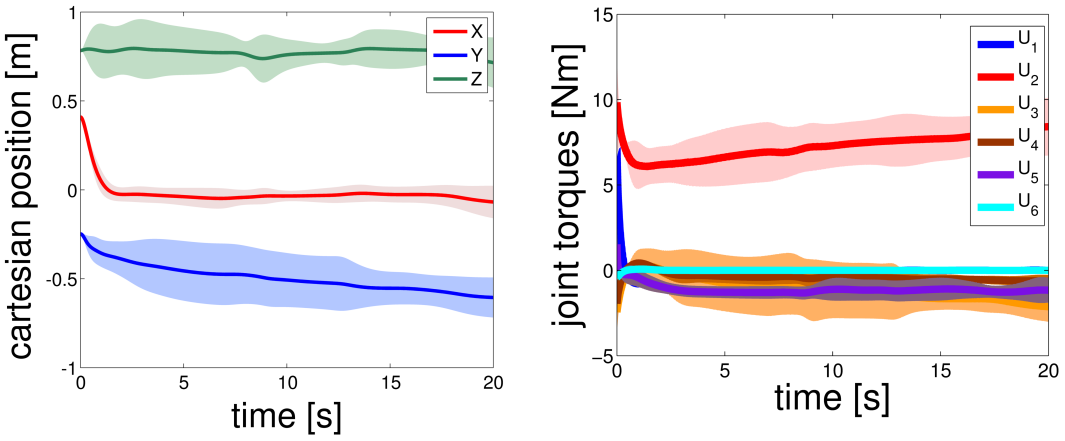
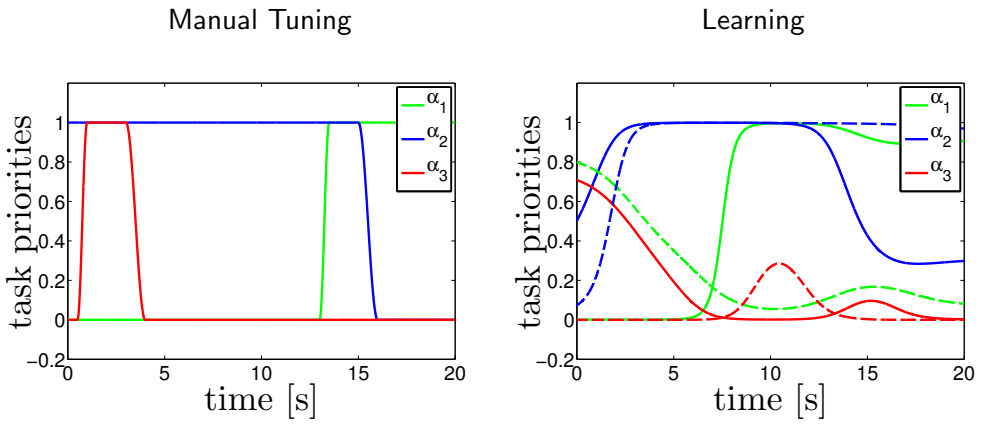


FIGURE 4.3: Mean and variance of the Cartesian trajectory of the end-effector and the joints torques of the simulated Jaco arm, generated by learned task priorities over 100 trials of the 3-tasks experiment (see text in Section 4.1.2). Even starting from random initialization of the task weight parameters, the learning process is eventually able to produce similar optimized motions of the robot that fulfil its 'global' task.

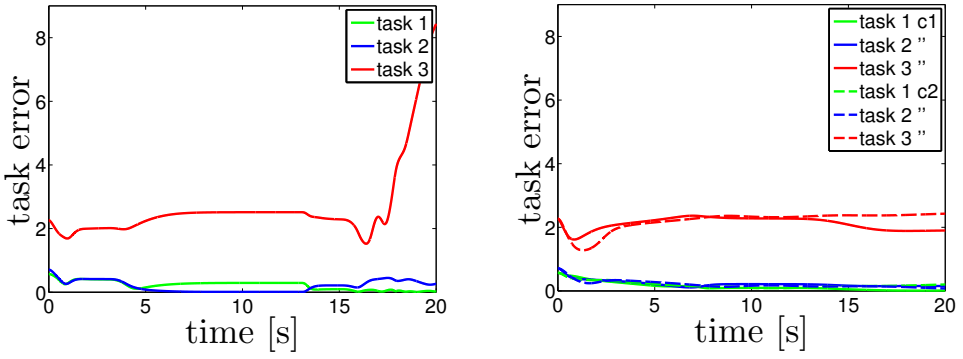
torques. Despite the redundancy of the robot and the one of the task priorities, the final robot movements are smooth and quite consistent with each other. Their average fitness is -0.0874 ± 0.0213. Overall, this result indicates that learning the soft task priorities starting from scratch (*i.e.*, where an initial guess for the activation of the task priorities in time is not available) is a viable and robust option for generating the motion of redundant robots.

4.1.3 Experiment on the Real Kinova Jaco Arm

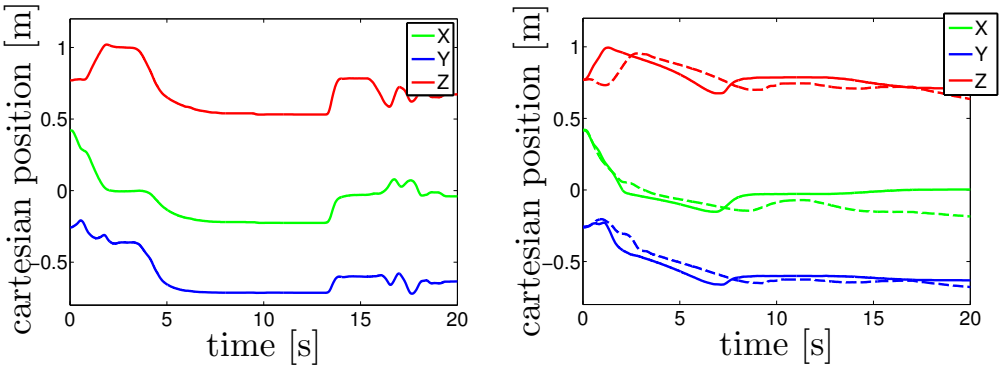
We compare in Fig. 4.4 the effect of three different task prioritizations on the real Jaco arm. In the left column, we show the robot movement generated by task priorities that were manually tuned by an expert user of the Kinova arm; on the right column, we show two typical robot movements generated by learned task priorities, which were optimized with CMA-ES starting from a known initial value (the manually tuned task weight functions) and a random value. We set the exploration rate in CMA-ES to 0.5 and perform 40 generations. Learning the priorities has a beneficial effect on the smoothness of the trajectories, which becomes evident when comparing the plots of the end-effector (Fig. 4.4c), joints positions (Fig. 4.4d) and torques (Fig. 4.4e) and the task errors (Fig. 4.4b). We evaluate the fitness using the commanded joint torques $\mathbf{u}_i$ and the kinematics and dynamics model of the Jaco arm to compute $\mathbf{p}_i$. The fitness value for the manually tuned task priorities is -0.1431. The fitness values for the two optimized solutions are better: -0.0585 and -0.0644 initializing the parameters with fixed and random values respectively. Overall, this experiment illustrates that learning the task



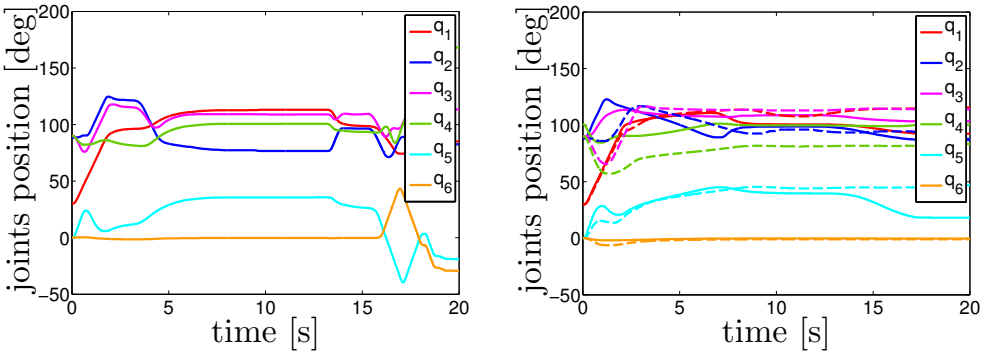
(A) The task priorities evolving in time.



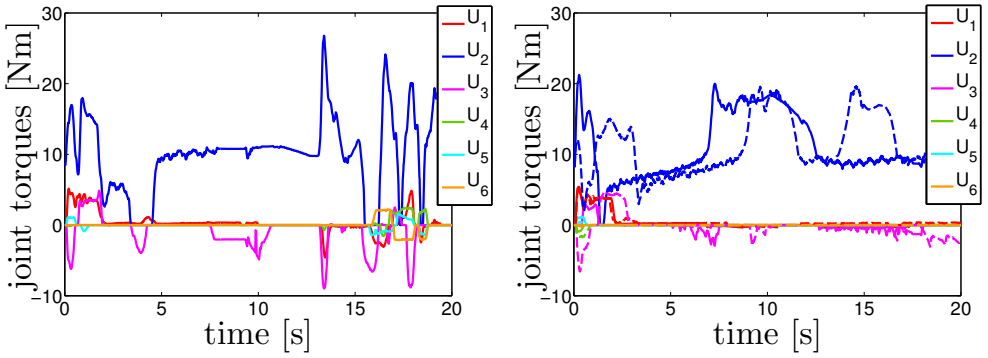
(B) The task errors.



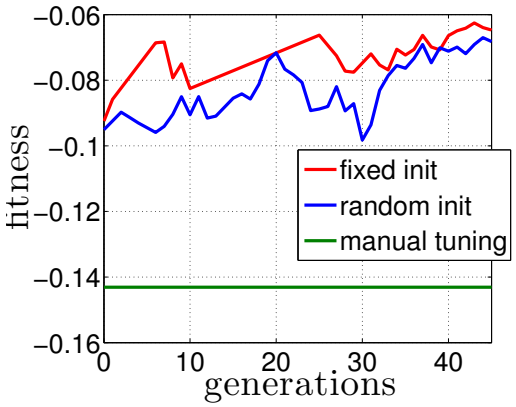
(C) The end-effector trajectory in the Cartesian space.



(D) The joints trajectories.



(A) The measured joints torques (estimated by the motor currents).



(B) The fitness values.

FIGURE 4.4: Comparison between a manually tuned (left side) and two typical learned (right side) policies for the 3-tasks experiment performed with the real Kinova Jaco arm. On the right, a solid line corresponds to a policy optimized starting from a fixed/known initial value of the priorities (*fixed init*), in this case the priorities found by manual tuning; the dashed line corresponds to a policy optimized starting from random values (*random init*). The final fitness values are: -0.1431 (manual tuning), -0.0585 (fixed init) and -0.0644 (random init).

priorities improves the real robot motion with respect to an existing manually tuned solution.

4.1.4 Comparison with the State-of-the-Art GHC

In this experiment we compare the performance of the task priorities learning applied to our method and to the state-of-the-art multi-task controller GHC [19].

In the GHC, each task is associated to a null space projector of the extended Jacobian that contains the analytical description of all the task objectives. Soft task prioritization is achieved because the null space projector depends on a set of manually designed weight functions ranging from 0 to 1, that control if each task is fully or partially projected in the null spaces of the other tasks with higher priority. The controller is the solution

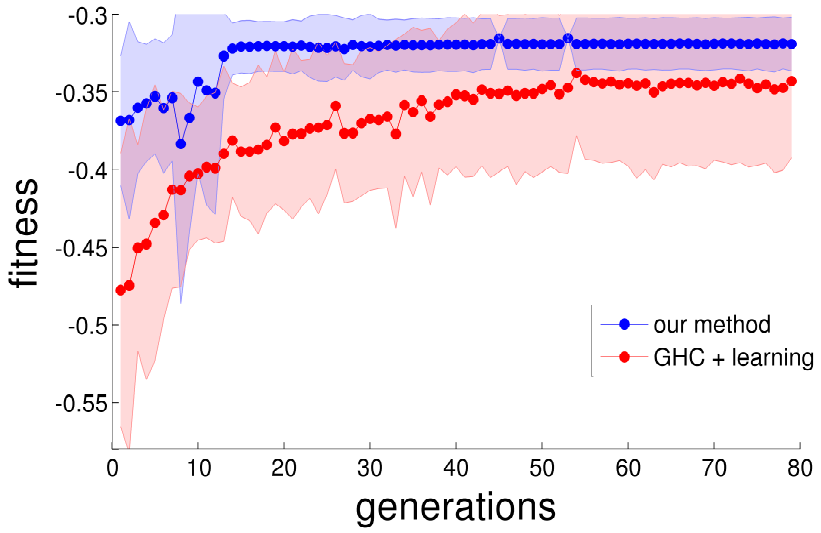


FIGURE 4.5: Comparison between our method and the GHC modified to learn the task priorities with CMA-ES. The plot shows the mean and the standard deviation of the fitness in $R = 20$ trials of the experiment with the simulated KUKA LWR arm (see Section 4.1.4). For both controllers, the learning is initialized with random parameters. Our method shows a faster convergence and better optimization of the fitness. The average fitness is is -0.0373 ± 0.0320 for our method and -0.0735 ± 0.0946 for the GHC+learning. The two distributions are statistically different ($p < 0.01$ with the K-S test).

to a quadratic optimal control problem subject to the robot and task constraints – see [19]. The soft task priorities are introduced as a further constraints, formulated by $\ddot{\mathbf{q}} = \sum_{i=1}^{n_t} \mathbf{P}_i(\mathbf{\Lambda}_i)\ddot{\mathbf{q}}_i^{'}$, where $\mathbf{P}_i(\cdot)$ is the null space projector associated to the i-th task, $\mathbf{\Lambda}_i$ is a matrix that depends on the task priorities, $\ddot{\mathbf{q}}_i^{'}$ are intermediate joint accelerations associated to each task and $\ddot{\mathbf{q}}$ are the joints accelerations. To enable the comparison with our method, we parametrized the task priority matrix $\mathbf{\Lambda}_i$ for each task i in the same way as described in Section 3.2.2.

We compare the two methods on a reaching task with a simulated 7 DoF Kuka LWR, which was originally used in [19]. In this scenario, the robot must reach a goal point beneath a rectangular surface parallel to the ground ($z = 0.25m$), without collision. The are 2 elementary tasks. The first is about reaching the Cartesian pose $\mathbf{p}^*$ =[0.6, 0, 0.15] with the end-effector (*goal*). The second is about keeping the joint configuration [0, 90, 0, -90, 0, 90, 0] (degrees). We design the following fitness function $\phi \in [-1, 0]$:

$$\phi(\mathbf{q}_{1,\dots,T}, \mathbf{u}_{1,\dots,T}) = -\frac{1}{2}\left(\frac{\sum_i^T \|\mathbf{p}_i - \mathbf{p}^*\|}{\epsilon_{\max}} + \frac{\max(\|\mathbf{u}_{i=1,\dots,T}\|_\infty)}{u_{\max}}\right)$$

where $\|\cdot\|_\infty$ is the infinity norm. We set the fitness to -1 in case of collision. We run 20 experiments from random initial parameters for both methods, with an exploration rate

of 0.1 for CMA-ES and 80 generations. Our method generated solutions that satisfy the collision constraint in 90% of the cases, while the GHC succeeded only in 75%. Figure 4.5 shows the mean and standard deviation of the fitness: our method is faster in convergence and improves the final optimized fitness. The average fitness at the end of the learning process is -0.0373 ± 0.0320 for our method and -0.0735 ± 0.0946 for the GHC+learning. The two distributions of the fitness are statistically different ($p = 0.0073 < 0.01$, obtained with the two-sample Kolmogorov-Smirnov test). Our method is also 10 times faster in terms of computational time: on a standard i7 machine, the average time for the optimization process to find a solution with 80 generations (average over 20 trials) is $3.7 \times 10^3 \pm 2.4 \times 10^2$ seconds for our method and $3.9 \times 10^4 \pm 2.2 \times 10^3$ seconds for the GHC+learning approach.

4.2 Optimizing Soft Task Priorities with Constraints

In this section, we apply (1+1)-CMA-ES with covariance constrained adaptation to our multi-task control framework (Section 3.2). We use it to optimize the task priorities and to obtain a solution that never violates the constraints. In the following, we report on the experiments performed to optimize the whole-body movements of the iCub humanoid robot.

We designed two experiments using the 17 DoF of the upper-body of the robot (arms and torso). In the experimental scenario, a rectangular obstacle similar to a wall, that is as large as the robot's chest and 2 *cm* thick, is placed about 20 *cm* in front of the robot.

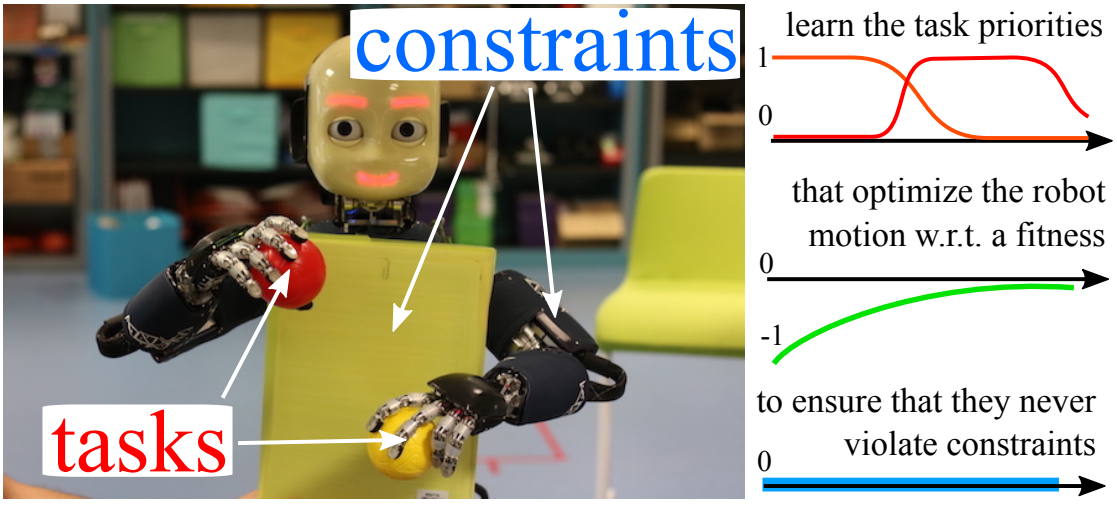


FIGURE 4.6: The humanoid robot iCub performing a bimanual task with several tasks and constraints. In this experiment we optimize the task priorities in a way that it never violates any of the constraints.

The first experiment is aimed at reaching a goal Cartesian position behind the wall with one hand. There are three elementary tasks. The first is about reaching the desired Cartesian position $\mathbf{p}_r^* = [0.35, -0.15, 0.7]$ (m) with the right hand frame of the robot. The second task is reaching a desired Cartesian position $\mathbf{p}_{elbr}^* = [0.24, -0.23, 0.7]$ (m) with the elbow frame. The third task is keeping the initial joint configuration $\mathbf{q}^* = [0, 45, 0, 0, -20, 30, 0, 0, 45, 0, 0, 0, 30, 0, 0, 0, 0]$ (deg). In sum, the goal is hidden behind the wall, and to reach it with the hand the robot must bend its elbow around the wall corner; the third task should prevent the robot from moving the right arm and the torso. The task priorities are approximated by RBFs with $n_r = 5$, therefore $n_P = 5 \times 3 = 15$. There are $n_C = n_{IC} = 73$ inequality constraints: joint position limits, joint torque limits and distance constraints to avoid collisions between the robot and the obstacle. Precisely, a minimal distance of 3 cm is required between the obstacle and a set of pre-defined collision check points (located at the origin of the frames of right shoulder, elbow, wrist, hand and head). For this experiment we use the following fitness

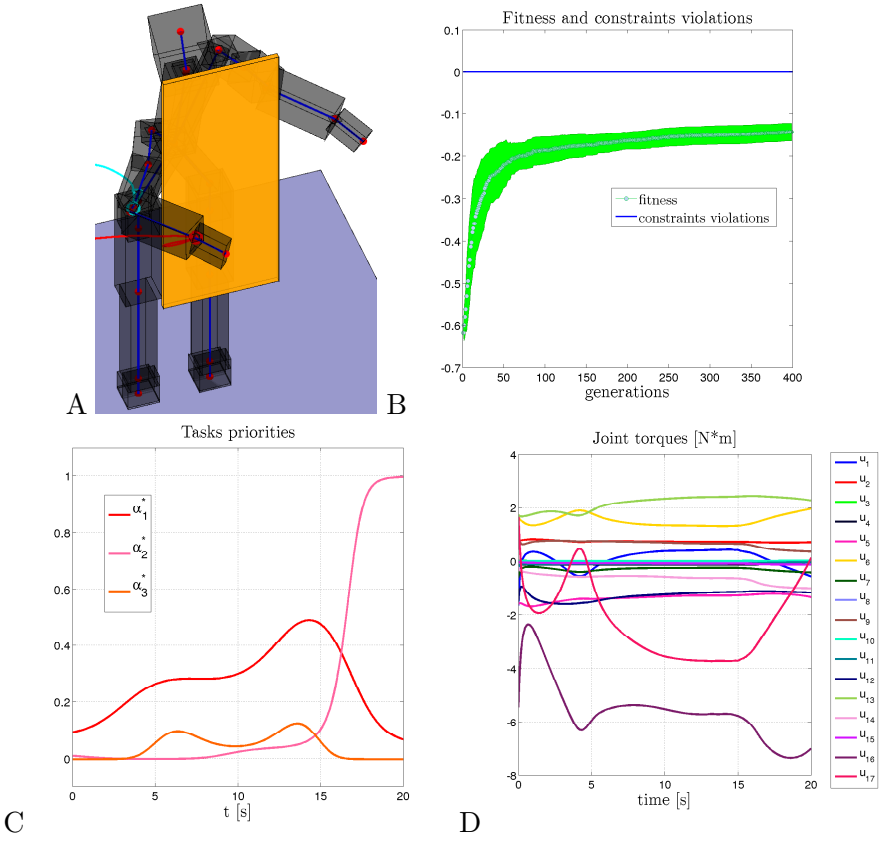


FIGURE 4.7: Experiment with the iCub, about reaching a goal behind the wall with one hand. A) The robot's movement visualized by the mex model. B) The median constraint violation and fitness optimized by (1+1)-CMA-ES with covariance constrained adaptation (over 25 experiments) the constraints are never violated C-D. The task priorities and joint torques of the best solution.

function:

$$\phi = -\frac{1}{2}\left[\frac{\sum_i^T \|\mathbf{p}_{r,i} - \mathbf{p}_r^*\|}{\varepsilon_{max}} + \frac{\sum_i^T \mathbf{u}_i^2}{u_{max}}\right] \tag{4.1}$$

where $\phi \in [-1, 0]$, T is the number of control steps (the task duration is 20 s, and we control at 1 ms), $\mathbf{p}_{r,i}$ is the right hand frame position at time i, $\mathbf{p}_r^*$ the goal position for the hand frame and $\varepsilon_{max} = 120$ and $u_{max} = 3.5 * 10^5$ are two scaling factors. The first term of ϕ penalizes the cumulative distance from the goal, while the second term penalizes the global control effort.

The second experiment complicates the first by adding 2 more tasks. The aim is to reach a Cartesian goal position with both robot hands. Two Cartesian goal tasks for each hand and elbow are set symmetrically with respect to iCub's sagittal plane. A fifth posture task is set as to keep the torso as straight as possible during the movement.

- Task 1 : $\mathbf{p}_r^* = [0.35, -0.15, 0.68]$ (m)

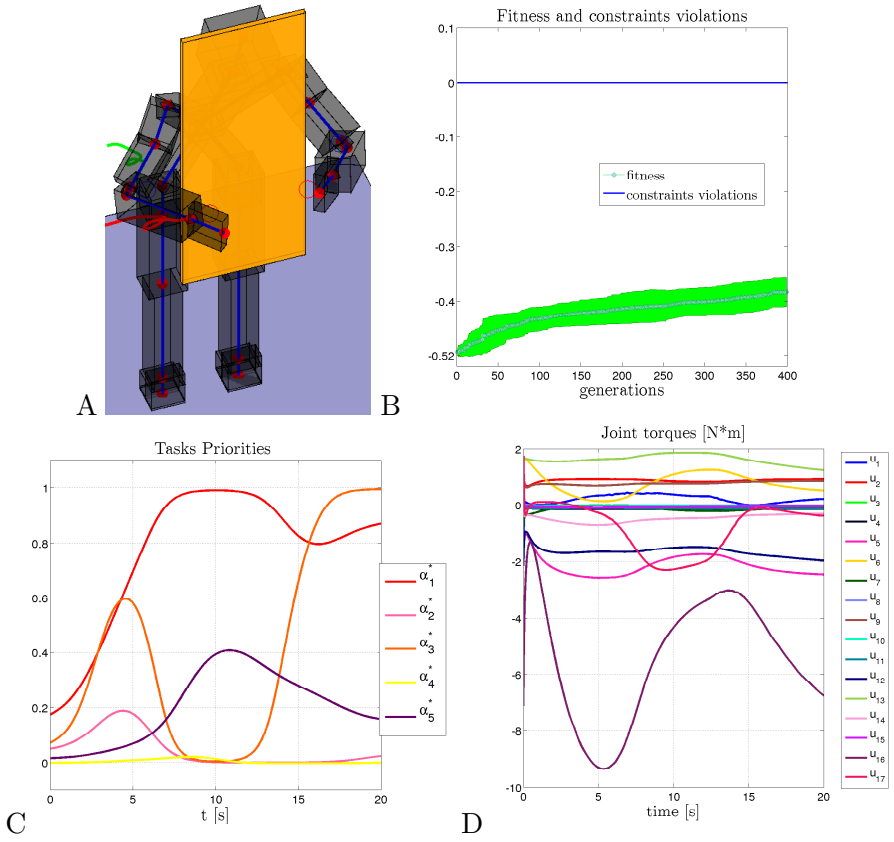


FIGURE 4.8: Experiment with the iCub, about reaching a goal behind the wall with two hands. A) The robot's movement visualized by the mex model. B) The median constraint violation and fitness optimized by (1+1)-CMA-ES with covariance constrained adaptation (over 25 experiments) the constraints are never violated C-D. The task priorities and joint torques of the best solution.

- Task 2 : $\mathbf{p}^*_{elbr} = [0.21, -0.25, 0.68]$ (m)
- Task 3 : $\mathbf{p}^*_{l} = [0.3, 0.0248, 0.68]$ (m)
- Task 4 : $\mathbf{p}^*_{elbl} = [0.21, 0.1138, 0.68]$ (m)
- Task 5 : $\mathbf{q}^* = [0, 45, 0, 0, -20, 30, 0, 0, 45, 0, 0, 0, 30, 0,\ 0, 0, 0]$ (deg)

The task priorities are approximated by RBFs with $n_r = 5$, therefore $n_P = 5 \times 5 = 25$. The optimization is carried out under the same constraints as in the first experiment with the addition of the left arm collision checks. This means we have $n_C = n_{IC} = 77$ inequality constraints. The fitness is:

$$\phi = -\frac{1}{2}\left(\frac{\sum_i^T \|\mathbf{p}_{r,i} - \mathbf{p}^*_r\|}{\varepsilon_{max}} + \frac{\sum_i^T \|\mathbf{p}_{l,i} - \mathbf{p}^*_l\|}{\varepsilon_{max}} + \frac{\sum_i^T \mathbf{u}_i^2}{u_{max}}\right) \tag{4.2}$$

where $\mathbf{p}_{l,i}$ is the left hand frame position at time i, $\mathbf{p}^*_l$ the goal position for the left hand frame.

In all the experiments, we seek the best solutions that do not violate any of the constraints. We employ (1+1)-CMA-ES as described in Section 3.6.3 with the exploration rate set to 0.1 (this is the only parameter to tune and this is the default value!).

Fig. 4.7 B and Fig. 4.8 B show the median fitness and constraint violation obtained by 25 experiments. The fitness grows nicely ($\phi = 0$ would be the optimum). Most importantly, the **constraints are never violated**, which is exactly what we wanted to obtain. We also show the task priorities and the joint torques from one of the best solutions; they are both smooth, and it is clear that optimizing the task priorities manually would be very difficult if these solutions were to be achieved.

4.3 Optimizing Whole-Body Trajectories

In this section we present our experiments with the iCub humanoid performing a "*stand-up from the chair*" movement. Our method is capable of optimizing the task trajectories to generate a safe motion even when the robot is switching contacts in physical interaction with the environment.

Robotics setup - Our experiments were performed with the iCub humanoid robot. iCub is a 53-DoF humanoid robot [78], but only 23-DoF are torque-controlled and used for balancing tasks. It is equipped with 6 force/torque sensors, placed in the middle of arms, legs and feet, that are used to compute the whole-body dynamics [79]. We developed our controller in Matlab/Simulink using the WBToolbox [80], which can be applied to

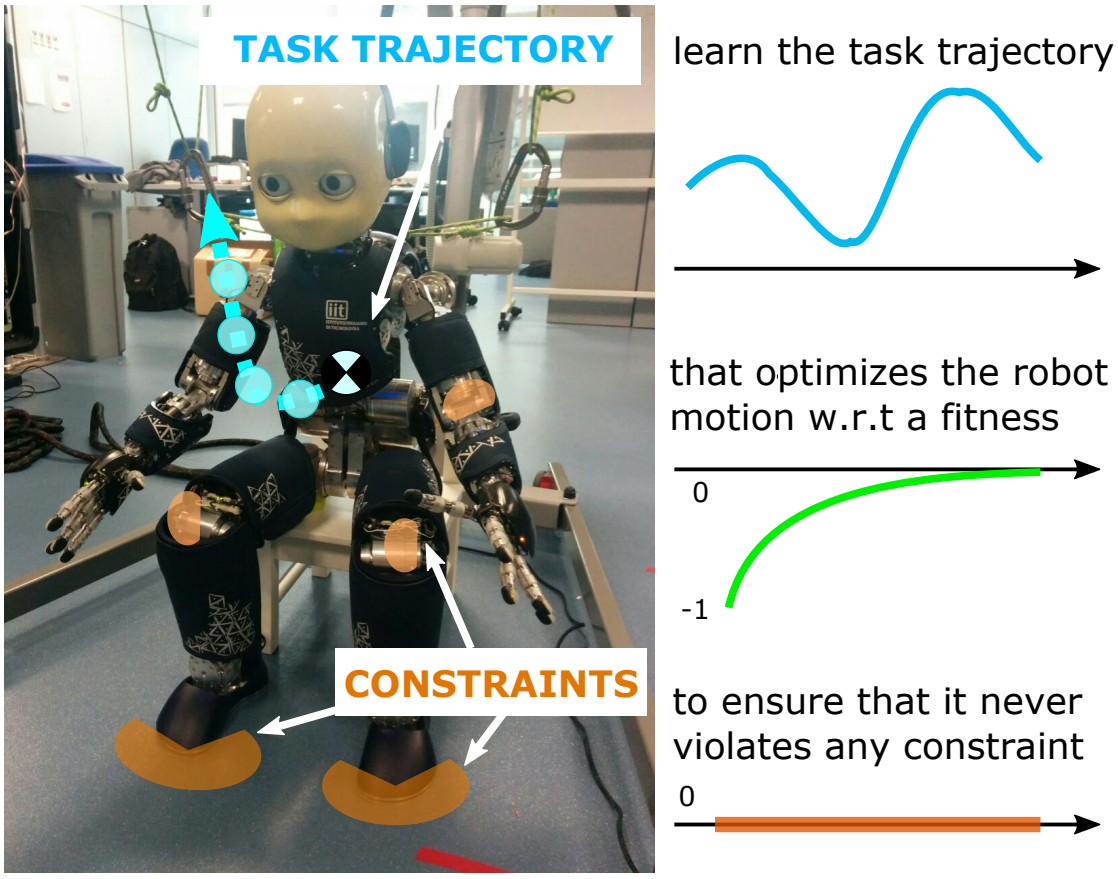


FIGURE 4.9: The humanoid robot iCub performing a whole-body motion (*stand-up from the chair*), with several tasks and constraints. In this experiment we optimize the desired task trajectory w.r.t. a fitness function, guaranteeing that the global robot behavior is safe: it never violates any of the constraints.

both the simulated and real robot, whereas the learning and trajectory optimisation is done with our framework for learning task priorities [31]. Considering the high chance of breaking the real robot during learning, we perform all the learning procedure in simulation. The rollouts are performed in Gazebo.

Stand-up problem - We apply our method to solve the problem of finding a safe trajectory for the robot to stand up from a chair, where "safe" means physically feasible and that it does not violate any problem/robot constraint. The goal of the experiment is to learn an optimal CoM trajectory to move the robot from a sitting position to a double support stance that guarantees constraints satisfaction and minimizes the risk of falling. The desired trajectory is executed by the centroidal momentum controller described in Section 3.3.1. For this experiment we constrain the desired CoM trajectory to lay on the sagittal plane of the robot (x, z plane in the robot world frame). For each trajectory component, we associate a RBFs network with $n^f = 5$ and we add two fixed viapoints for the starting and the final CoM positions. Then we use a third RBFs network to model the time law. To allow for the stand up movements in the controller, we need to

Posture	Torso (deg)	Larm (deg)	Lleg (deg)
$\mathbf{q}_{si}$	60, 0.62, 0.40	-67.2, 34.1, 4.8, 43.2	84.3, 0.8, 0.1, -99.2, -15.8, 0.1
$\mathbf{q}_{si}$	-10, 0, 0	-20, 30, 0, 45	25.5, 0, 0, -18.5, -5.5, 0

TABLE 4.1: The intermediate ($\mathbf{q}_{si}$) and the final joint pose ($\mathbf{q}_{st}$) that define the secondary task.

switch from the lower back/legs contact points (when the robot is sitting on the chair) to the feet contact points. Therefore we introduce a switching time t_{switch}. This induces a natural segmentation of the movement in two distinct phases: the first is when the robot is sitting, the second is when the robot breaks the contacts with the chair and starts to stand up to reach the standing position. For the posture task we set two different keyframe joint postures: one at the end of sitting phase ($\mathbf{q}_{si}$) and one a the end of the sit up movement ($\mathbf{q}_{st}$).

To ensure a smooth transition in the joint space, we used the method from [81]. To assure the feasibility of the optimal solution and to lighten the burden of the deployment on the real robot we introduce a set of constraints. For this experiment we set $n_C = n_{IC} = 93$ inequality constraints: joint position limits, joint torque limits and a dynamic balance constraint to assure that after the t_{switch} the Zero Moment Point (ZMP) is located inside the support polygon of the robot.

4.3.1 Optimization for the Simulated iCub in Gazebo

To obtain an optimal CoM trajectory that never violates the constraints we apply (1+1)-CMA-ES with CCA to our framework. As described in Section 3.6.3, to avoid stalling, the algorithm requires a starting point that verifies the constraints, but for our application scenario it is not straightforward to set the right parameters that verify all the constraints at once.

Therefore we split the optimisation problem in two. In the first, we run an unconstrained optimisation problem, where we want to minimize the constraints violation to 0 (no constraint violation). In the second, we run a constrained optimization problem to find a solution that satisfies our "true" cost function. The two problems have similar settings ($t_{switch} = 1.5$s, 3 RBFs networks with 5 RBFs each, for a total number of parameters equal to 15, $T = 4.5$s with a control step of 10ms) but different fitness functions.

The fitness function for the first problem, or bootstrap problem, (*unconstrained optimization*) is:

$$\phi_{uc} = \begin{cases} \hat{l}, & \text{if fallen} \\ \phi_{bs}(T, n_c) & \text{otherwise} \end{cases}$$

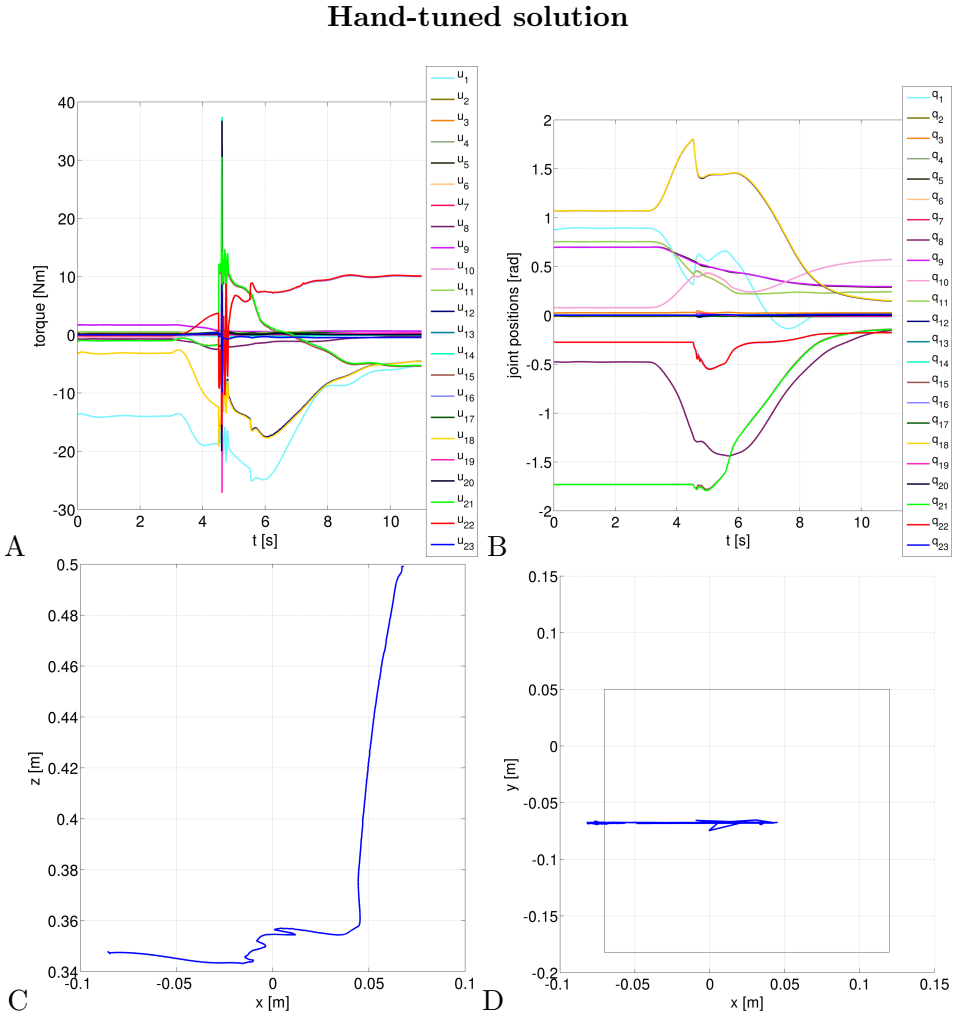


FIGURE 4.10: Typical trajectory that is manually hand-tuned by an expert. A plots the joint torques, B plots the joints positions during the experiment, C shows the CoM trajectory of the robot on the sagittal plane, D shows the evolution of the ZMP on the ground plane. In D, the square is a conservative estimation of the support polygon of the robot: for this reason, even if the ZMP of the hand-tuned experiment moves outside the support polygon, the robot does not fall.

where ϕ_{bs} is the fitness function from Section 3.7.4.2, and $\hat{l}$ is a penalty that we apply when the robot falls.

The fitness function for the second problem, (*constrained optimization*) is:

$$\phi_{co} = \begin{cases} \hat{l}, & \text{if fallen} \\ -\frac{1}{\sum_i^3 w_i}\left[\frac{w_1 \sum_i^T |\mathbf{p}_{com}(i) - \mathbf{p}^*_{com}|}{\varepsilon_{max}} + \frac{w_2 \sum_i^T \mathbf{u}^2(i)}{u_{max}} + \frac{w_3 \sum_i^T b(i)}{b_{max}}\right], & \text{otherwise} \end{cases}$$

where $\phi_{co} \in [-1, 0]$, w_i are fixed weights, $\varepsilon_{max}, u_{max}, b_{max}$ are normalization factors for, respectively, the CoM position error (where $\mathbf{p}_{com}(i)$ is the current CoM position and $\mathbf{p}^*_{com}$ is the desired CoM), the effort penalty ($\mathbf{u}(i)$ is the torque at time i) and the

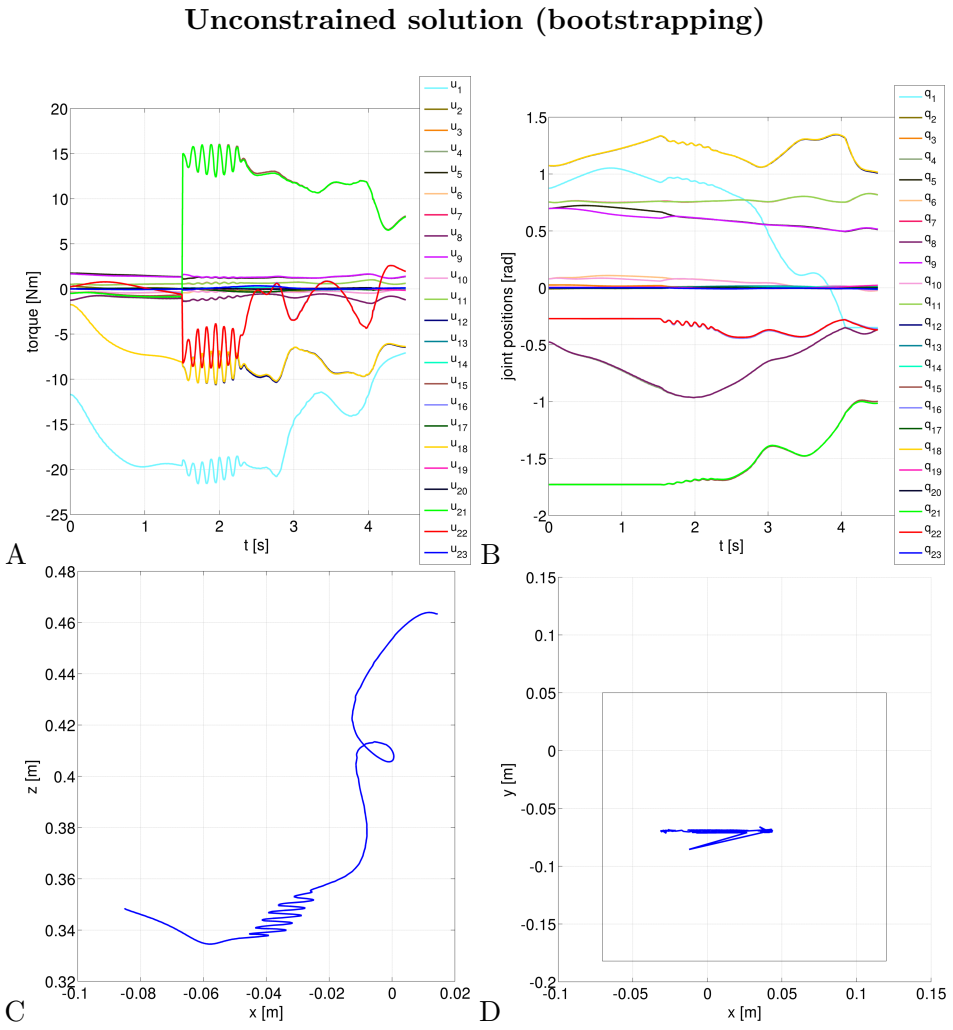


FIGURE 4.11: Typical solution after the bootstrapping. A plots the joint torques, B plots the joints positions during the experiment, C shows the CoM trajectory of the robot on the sagittal plane, D shows the evolution of the ZMP on the ground plane. In D, the square is a conservative estimation of the support polygon of the robot.

backward penalty. The latter is defined as

$$b(i) = \mathbb{1}_{\{\mathbf{p}^x_{com}(i-1) > \mathbf{p}^x_{com}(i)\}}(\mathbf{p}^x_{com}(i-1) - \mathbf{p}^x_{com}(i)) + \tag{4.3}$$

$$\mathbb{1}_{\{\mathbf{p}^z_{com}(i-1) > \mathbf{p}^z_{com}(i)\}}(\mathbf{p}^z_{com}(i-1) - \mathbf{p}^z_{com}(i)) \tag{4.4}$$

where p^x_{com}, p^z_{com} are the x and z coordinate of the CoM position. This term penalizes all the trajectories that produce backward movements along the x or z direction. We use it to minimize undesirable oscillations of the robot along the sagittal plane.

To find a solution for the first problem (bootstrap) we performed 500 rollouts of (1+1)-CMA-ES (without CCA) with an exploration rate equal to 0.1. The solution is used as the starting point for the second problem, which is solved with (1+1)-CMA-ES with CCA, with the same exploration rate. A typical fitness profile for ϕ_{co} after 500 rollouts

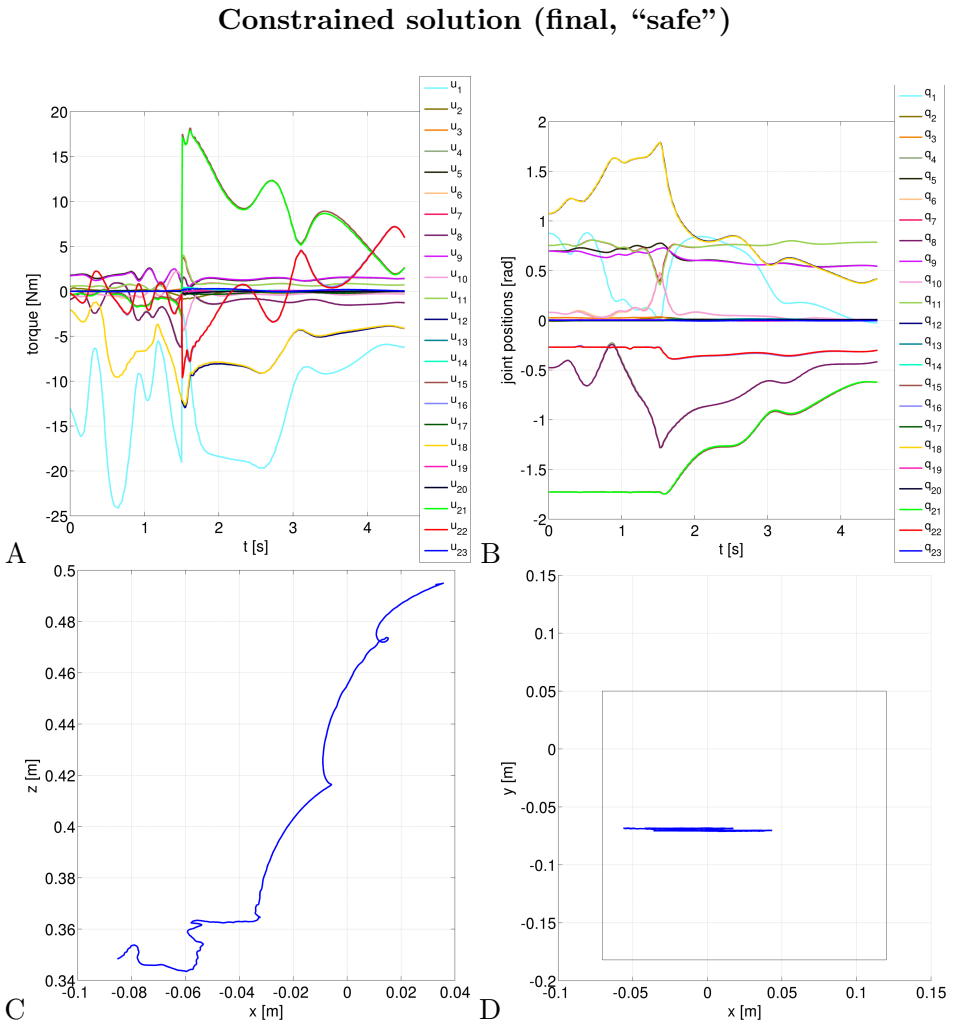


FIGURE 4.12: Typical final solution obtained by the constrained optimization. A plots the joint torques, B plots the joints positions during the experiment, C shows the CoM trajectory of the robot on the sagittal plane, D shows the evolution of the ZMP on the ground plane. In D, the square is a conservative estimation of the support polygon of the robot. Our optimization algorithm finds solutions that satisfy the conservative ZMP constraint.

is shown in Figure 4.13. For this experiment, the weights for the penalty terms are respectively $w_1 = 1.3$ for the CoM error, $w_2 = 1$ for the effort and $w_3 = 3$ for the backward penalty.

Solution	CoM error	effort penalty	backward penalty
hand-tuned	1.0	1.0	0.186292
unconstrained	0.806484	0.945829	0.962220
constrained	0.580518	0.646327	0.844451

TABLE 4.2: Comparison of three typical solutions: hand-tuned, unconstrained (used for bootstrapping) and the final found by the constrained optimization. For each solution we separately compute each cost that composes the fitness ϕ_{co} in Equation 4.3.1. The constrained solution computed at the end of the whole optimization process has a better performance than the hand-tuned both in terms of COM error and total effort.

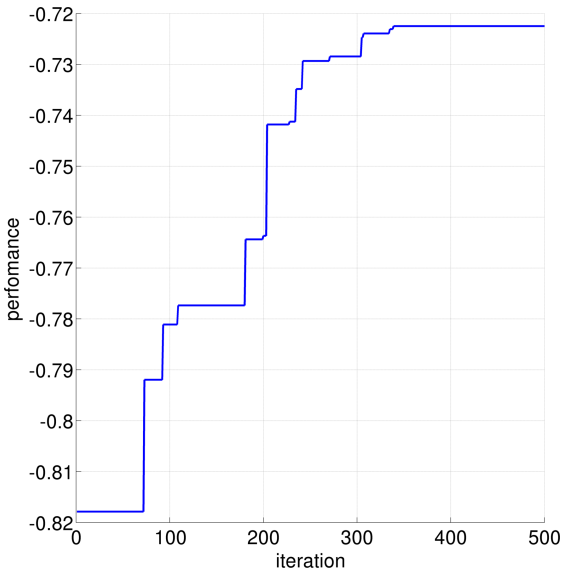


FIGURE 4.13: Evolution of a typical fitness value for ϕ_{co}.

We compare the optimized solution with one that was hand-tuned by an expert. The hand-tuned solution lasts for 11s, with $t_{switch} = 4.53$s. Even if the hand-tuned solution has a smoother profile for the CoM, it makes the robot move faster during the stand-up and with higher torques (with a peak of torques at t_{switch}). The optimized trajectory, on the contrary, is faster (less than 5s), has lower energy consumption and lower CoM tracking error, as shown in Table 4.2. By design, our solution satisfies all the problem constraints: as shown in Figure 4.12, it satisfies a very conservative constraint on the ZMP, that the hand-tuned solution was violating even if it was generating a physically feasible movement.

Chapter 5

Conclusions

In this thesis we have shown that better results stem from the combination of machine learning approaches with torque control schemes, whereas, using solely one of the aforementioned approach, lacks of robustness and flexibility. This paradigm inherit from the machine learning the capability of providing optimal solutions for problems that due of complexity lack of a precise mathematical models, while it gets from the control theory the robustness and the repeatability that are highly desirable for real application scenario. therefore in this thesis, we propose a novel framework located at the confluence of machine learning and control theory that is extremely versatile and tackles a broad range of different problems. Originally we started by addressing an important issue for prioritized multi-task controllers, that is the automatic and optimal generation of task priorities through parametrized weight functions. As a first step towards an automatically tuned controller for redundant robots, we propose an adaptation of our framework with a multi-task controller where the task priorities can be learned via stochastic optimization. We show the effectiveness of our approach by comparing to GHC [17], a state-of-the-art multi-task prioritized controller. We present several results performed on a simulated 7 DOF Kuka LWR arm and both a simulated and a real 6 DOF Kinova Jaco arm.

From the results collect in this first stage we realize that constraints satisfaction plays a major role in the automatic search of an optimal solution. We noticed that with a simple constraints management, based on a death penalty criteria for the unfeasible solutions, we easily got stuck during the optimization process.

Therefore, in our second work, we proposed to optimize the task priorities of multi-task controllers by a stochastic *constrained* optimization algorithm that ensures that the constraints were never violated. We benchmarked four constrained optimization algorithms on analytical functions and robotics applications and found that (1+1)-CMA-ES

with covariance constrained adaptation meets our requirements in terms of fitness of the solution and constraints satisfaction. In this work we showed that our framework is capable of generating optimized whole-body movements that always comply with safety requirements. As shown in two bimanual experiments with the iCub both in simulation and on the real robot, our method compute solutions that can be easily deployed on the real robot with reduced risk of damaging the robot due to our constraints satisfaction requirements In this paper we propose a method to compute safe task trajectories for whole-body control of humanoid robots, where "safe" means that we ensure that the optimized trajectories lead to behaviours that never violate any of the problem/robot constraints. In our last work we extend our framework to trajectory optimization problems. Here we proposed a method to compute safe task trajectories for whole-body control of humanoid robots, where "safe" means that we ensure that the optimized trajectories lead to behaviours that never violate any of the problem/robot constraints. We propose to use (1+1)-CMA-ES with CCA, that we previously benchmarked in [31], to optimize the parameters of the task trajectories, while for controlling the robot we rely on a whole-body multi-task centroidal momentum controller from [22, 23].

We demonstrate our method on the iCub humanoid, to find a safe trajectory for the motion "stand-up from the chair", with multiple switching contacts and several constraints. By comparing the optimal solution with an hand-tuned one we showed that our method improves the task trajectories satisfying all constraints and reducing the energetic consumption. Realizing this kind of motions was not possible in our previous framework where task priorities were optimized [31].

Even if our method showed great robustness in addressing different problems and strong efficacy in finding optimal solutions it has several drawbacks. The first issue that we want to address refer to the time to find an optimal solutions. The optimization engine of our method, CMA-ES, suffers from a slow converging rate. In this manuscript we propose a draft solution to tackle this issue. We designed a novel version of (1+1)CMA-ES with CCA that harnesses the capability of Bayesian Optimization scheme to minimize the number of trial to find an optimal solution. The algorithm is described in Chapter 3. In this work we present only preliminary results but they are quite promising and they provide good insight on how to solve this issue. The second issue that we want to tackle is referred to the lack of generalization that we experience when it comes to deploy our solutions on a real robot. Imposing the safety of our solution is a necessary but not sufficient condition to assure the one-shot deployment of our controller. Currently, the solution is optimized in simulation, as the number of roll-outs is too high for executing the algorithm directly on the real robot. However, the feedback controller and the good dynamics model make it applicable with minor adaptations to the real robot. In the future we plan to take inspiration from the "Intelligent Trial and Error" approach, which

essentially suggests to generate a large diversity of high-performing trajectories offline and select the best one to test on the robot online by trial and error [66].

Appendix A

Balance Criteria for Humanoids Robots

In this appendix we present a brief introduction of one of the most used balance criteria for humanoid robots. This description is based on the paper by Sardain and Bessonet [**?**]. In literature there exists many balance criteria that to establish if a certain robot internal configuration guarantees the stability of humanoid robots. In recent years scholars proposed many criteria such as Foot Rotation Indicator (FRI) [**?**], Centroidal Momentum Pivot (CMP) [**?**], Capture Point (CP) [**?**], zero rate of change of angular momentum (ZRAM) [**?**], etc. In this appendix we focus our attention on the first balance criteria introduced in 1969 by Vukobratovic and Juricic [**?**], the Zero Moment Point (ZMP) and its relationship with the center of pressure (CoP). In this summary we consider multi actuated joints system establishing only one rigid flat stable contact with the environment. In this circumstances there are two different type of forces acting on the humanoid: forces acting on the contact surface and contact-less forces such as gravity and inertia forces. Contact forces define CoP while ZMP depends upon the forces exerted without contacts. At the contact surface the pressure field (orthogonal to the surface) is equivalent to a single force with a zero moment application point, the CoP. On the other hand The ZMP is defined as the point on the ground where the component, tangential to the contact surface, of the moment exerted by gravity and inertial forces is equal to zero. The previous assertion means that the tangential moment could be different from zero. For this reason a more suitable definition for ZMP would be "zero tipping moment point". To compute the CoP we consider a generic point O on the contact surface and $\mathbf{n}$, a normal vector to the contact surface with unit norm. Given the wrench $\mathbf{R}_O^p, \mathbf{M}_O^p$ associated to the pressure forces at the point O and knowing the

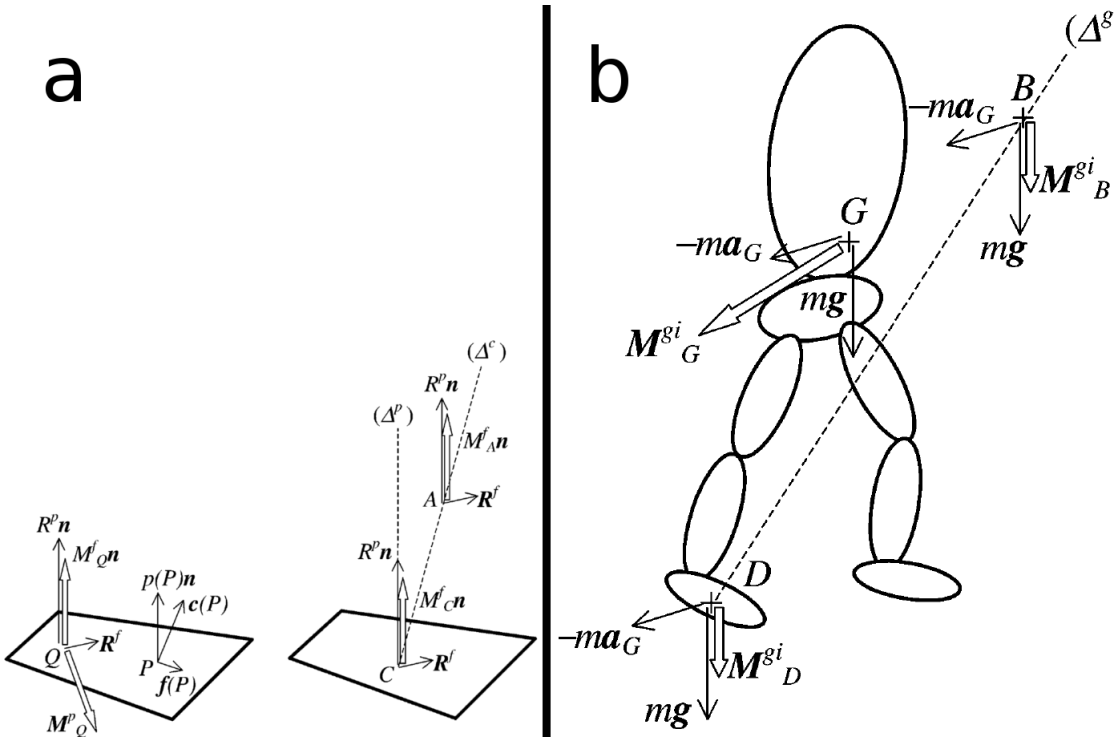


FIGURE A.1: In this figure we show the forces and moments locations at the contact surface (image a) and at the Center of Mass (image a). In the image a the term Δ^c represents the axis where the contact moment applied in each point of Δ^c is perpendicular to the surface of contact. The CoP is defined as the intersection of this axis with the contact surface. In the image a the Δ^g is th axis where the gravity and inertial forces moment is always parallel with the unit norm vector $\mathbf{n}$. The intersection of Δ^g with the contact surface identifies the ZMP location. This image is taken from [**?**].

fact that the tipping moment vanishes at the CoP we can compute the CoP location as:

$$\mathbf{M}^p_O = \mathbf{OC} \times \mathbf{R}^p \tag{A.1}$$

and by inverting the previous relation we obtain:

$$\mathbf{OC} = \frac{\mathbf{n} \times \mathbf{M}^p_O}{R^p} \tag{A.2}$$

it is easy to show that the Equation A.2 can be written as a function of the total contact forces acting at the point O as

$$\mathbf{OC} = \frac{\mathbf{n} \times \mathbf{M}^c_O}{\mathbf{R}^c \cdot \mathbf{n}} \tag{A.3}$$

where here the $(\cdot)^c$ apex means that we are considering the total contact forces and moments acting at the application and not only its orthogonal components to the surface (represented with a $(\cdot)^p$ apex).

For the ZMP computation we start from the the Newton-Euler equation of motion of the entire multi-articulated robotics platform

$$\mathbf{R}^c + m\mathbf{g} = m\mathbf{a}_G \tag{A.4}$$

$$\mathbf{M}^c_Q + \mathbf{QG} \times m\mathbf{g} = \dot{\mathbf{H}}_G + \mathbf{QG} \times m\mathbf{a}_G \tag{A.5}$$

where $\mathbf{g}$ is the gravity vector, m is the robot total mass, $\mathbf{a}_G$ is the acceleration vector of the robot Center of Mass G, $\dot{\mathbf{H}}_G$ is the rate of change of the angular momentum at the Center of Mass and $\mathbf{QG}$ is the vector joining a generic point at the contact surface Q with the Center of Mass G. The Equation A.4 can be rewritten as:

$$\mathbf{R}^c + (m\mathbf{g} - m\mathbf{a}_G) = 0 \tag{A.6}$$

$$\mathbf{M}^c_Q + (\mathbf{QG} \times m\mathbf{g} - \dot{\mathbf{H}}_G - \mathbf{QG} \times m\mathbf{a}_G) = 0 \tag{A.7}$$

Due to the fact that the wrench applied to the Center of Mass (represented with the $(\cdot)^{gi}$ apex) can be written has

$$\mathbf{R}^{gi} = m\mathbf{a}_G - m\mathbf{g} \tag{A.8}$$

$$\mathbf{M}^{gi}_Q = \dot{\mathbf{H}}_G + \mathbf{QG} \times m\mathbf{a}_G - \mathbf{QG} \times m\mathbf{g} \tag{A.9}$$

we can substitute the previous two expression inside the Equation A.6 and we obtain

$$\mathbf{R}^c + \mathbf{R}^{gi} = 0 \tag{A.10}$$

$$\mathbf{M}^c + \mathbf{M}^{gi} = 0. \tag{A.11}$$

This relations state that the humanoid stance presents a dynamic balance if the contact and the gravity-inertia forces are opposite. Due to this opposition the ZMP location, D, can be computed with an equation analogous to the Equation A.2

$$\mathbf{OD} = \frac{\mathbf{n} \times \mathbf{M}^{gi}_O}{\mathbf{R}^{gi} \cdot \mathbf{n}}. \tag{A.12}$$

From the last expression is possible to derive the more common definition of ZMP that is recurrent in the literature

$$\mathbf{OD} = \frac{mg\mathbf{z} \times \mathbf{OG} \times \mathbf{z} + \mathbf{z} \times \dot{\mathbf{H}}}{mg + m\mathbf{a}_G \cdot \mathbf{z}}. \tag{A.13}$$

This equation is true only if the ground is horizontal with $\mathbf{n} = \mathbf{z}$ and $\mathbf{g} = -g\mathbf{z}$. We know that if the Equation A.10 are satisfied, then the robot is dynamically balanced so in this it follows that the CoP and the ZMP are the same point. This equality does not hold when we are in a condition of dynamically unbalanced contact with the environment

(the robot is rotating around one of the contact surface edge). In this situation the ZMP does not represent a physical point and it is located outside the support polygon. When the ZMP leaves the support polygon it means the occurrence of a moment that cannot be counteracted by contact surface reaction forces. The distance of the ZMP from the support polygon provide an estimation of the unbalance moment, and even if we are in a dangerous situation that may bring the downfall of the humanoid it is still possible to define a correct dynamic reaction to bring back the ZMP inside the support polygon and re-establish a dynamic balance.

Appendix B

Experimental Platforms

In this thesis, for testing the proposed framework, we performed our experiments on two different platforms: a manipulator from the Kinova Robotics called Jaco and a humanoid robot entirely developed at the Italian Institute of Technology (IIT), the iCub.

The Jaco arm is a manipulator launched in 2010 by Kinova, a canadian company founded in 2006. The Jaco arm is a light weight manipulator (only 6 Kg) and it was originally designed for the assistance of people with upper limbs impairments. The manipulator has 6 Degree Of Freedom and is equipped with a three fingers gripper. Each link is made of carbon fibre to keep small the weight of the entire structure and each joint can be controlled in position or velocity and , only recently, in torque. The robotic arm has a maximum payload of 1.5 Kg and can reach objects at 90 cm from the robot. The reduced weight and the compact dimensions permits an easy deployment of the robot on every surface. The gripper consists of 3 underactuated fingers and each finger can be independently flexed. The Jaco arm can be controlled through a joystick that implements very simple primitive movements. The joystick control allows for simple cartesian position(x, y and z axis) and orientation (roll, ptich and yaw) of the end effectors. Moreover with the joystick is possible to control grasp and release of the hand with two or three fingers. The Jaco arm possess a rich set of API that allows for the control of the arm remotely. The Jaco arm is equipped with a usb 2.0 connection that make the connection with the control interface completely effortless. For this thesis we developed a controller interface for the kinova written in C++. To provide a feedback cartesian trajectory controller capabilities we employed the Open Kinova Driver Library [**?**] that are natively designed for multi-threading. In this way we where capable of measuring the state of the robot and compute the next control input with a 10 kilohertz control loop. The library comes with a visual interface (developed by Sebastian Marichal)that can be used to control the arm using the extended set of functions provided

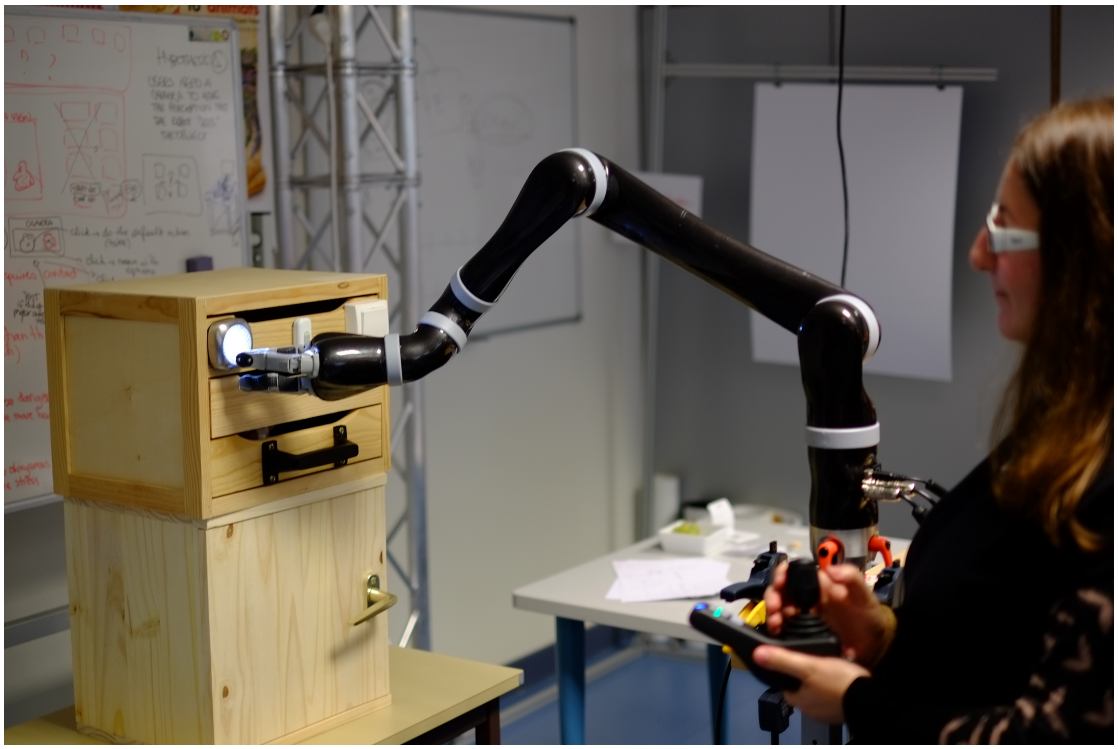


FIGURE B.1: A remotely controlled Kinova performing a manipulation task.

by the library. The library was employed to perform multi task learning [26] and for the interface evaluation in human robot interaction [**?**]. This library can be downloaded at https://github.com/serena-ivaldi/kinova-modules.

The iCub project was launched in 2004 by the RobotCub Consortium, an association of several European universities leaded by the IIT. The project has received 8.5 million Euro until it's end in 2010. The developed platform is entirely open-source and the entire documentation (hardware design and software) is released under a shareable license. The RobotCub project is based on the idea that developing a human like robot could push forward our understanding of the cognitive capabilities both natural and artificial and how they develop over time. The iCub platform [**?**] is 104 cm tall, is compact design (it weights only 22 kg) is suitable for a wide range of cognitive task like manipulation and general interaction with the environment. The great number of Degrees of Freedom (53) gives to the platform extreme flexibility to perform the most diverse tasks. The upper body has a total of 38 DoF, 7 for the arms, 9 for the hands and 6 for the head. To give to the platform the capability of performing many locomotions tasks such as walking, sitting or squatting, the designers provided the robot with 6 DoF for the legs with 3 at the hip, 1 for the knees and 2 DoF for the ankle. To extended the motion capability of the robot for manipulation and crawling 3 DoF in the waist were considered sufficient in order to reach a larger workspace for the upper body. For interacting with the environment the robot has to be equipped with exteroceptive and proprioceptive sensors. The

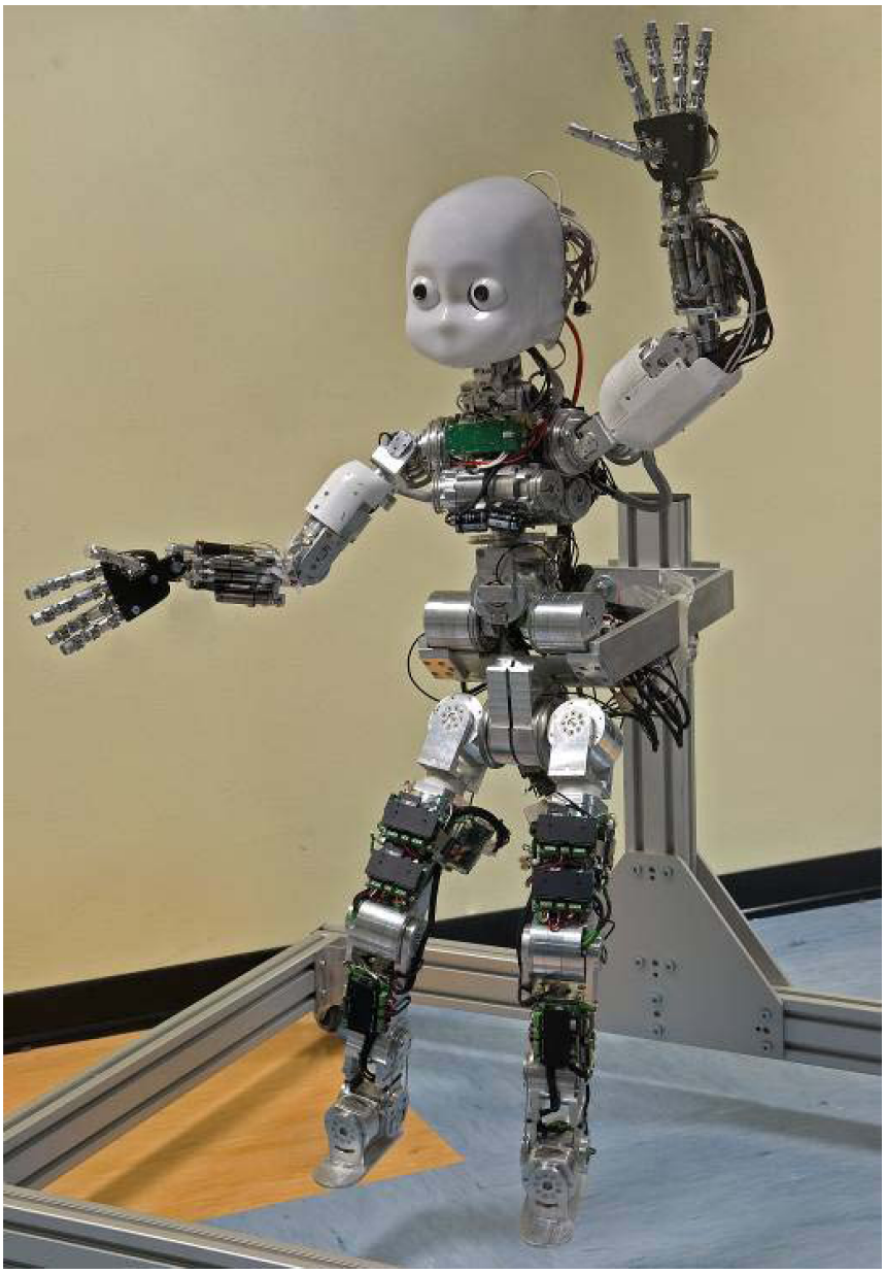


FIGURE B.2: In this figure we show an iCub prototype without its skin. Source: [**?**]

iCub can counts on a series of cameras, inertial sensors force-torques, position and tactile sensors that provide to the robot enough information to deal with the environment. The software modules that control the iCub are base on the Yarp system. The yarp system is a communication layer that acts as a transparent interface among algorithms and the hardware of the robot. Yarp introduces a set of protocols for inter-processing communication. Yarp defines *ports* and *devices*. The former can deliver massage across the Yarp network using several different protocols, the latter create an interface that provide all the devices capabilities through a set of APIs. A recently developed of the iCub software ecosystem is directed toward the integration of the Yarp system with Matlab-Simulink.

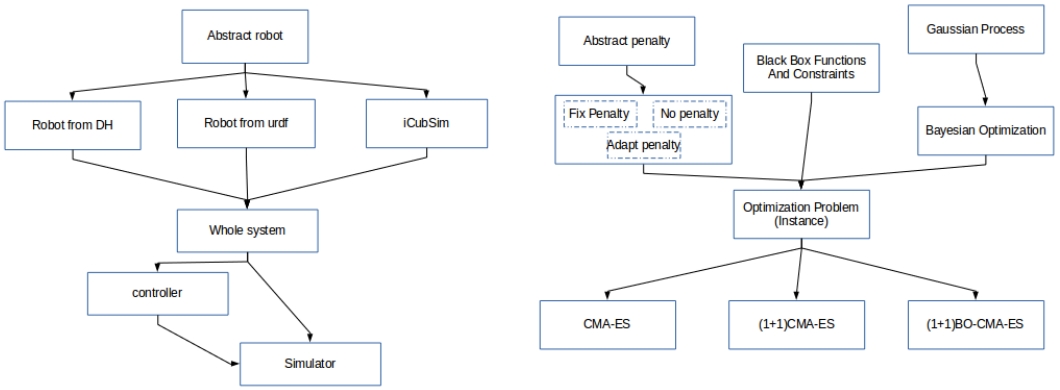


FIGURE B.3: In this figure we presents a conceptual scheme of the code that we developed for this thesis. On the left we show how we define an abstraction for the robot classes and how the controller class relates to them. On the right we show how we structure a black box optimization problem and the optimization method that we provide in the toolbox(in the picture we do not show all the variants of constrained CMA-ES associated with the different penalty methods that are available in the software package)

For the CoDyCo project the IIT Dynamic interaction Group released a toolboxes to directly design iCub controller inside the Simulink simulator. This simulink control schemes can be used to control both a simulation of the iCub and a the real platform with few changes of configuration parameters. One of the main contribution of our thesis is the integration of the iCub Simulink controller inside our framework. We developed a software system that provides an abstract interface for the simulation and control of robots (based on the Robotics Toolbox [56] from Peter Cork and whole body control toolbox [57] from IIT) and a set of routines for black box optimization (constrained and unconstrained) that is problem agnostic and requires simple wrapping procedure to be applied to new optimization problem. The framework is developed in Matlab and it is freely available for download at https://github.com/serena-ivaldi/learnOptimWBC.

Bibliography

[1] Nicolas Mansard and François Chaumette. Task sequencing for sensor-based control. *IEEE Transactions on Robotics*, 23(1):60–72, 2007.

[2] João Silvério, Sylvain Calinon, Leonel Dario Rozo, and Darwin G. Caldwell. Learning competing constraints and task priorities from demonstrations of bimanual skills. *CoRR*, abs/1707.06791, 2017.

[3] Alexandros Paraschos, Rudolf Lioutikov, Jan Peters, and Gerhard Neumann. Probabilistic prioritization of movement primitives. *IEEE Robotics and Automation Letters*, 2(4):2294–2301, 2017.

[4] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4):43:1–43:8, 2012.

[5] Y. Tassa, N. Mansard, and E. Todorov. Control-limited differential dynamic programming. In *ICRA*, 2014.

[6] Philipp Reist, Pascal Preiswerk, and Russ Tedrake. Feedback-motion-planning with simulation-based lqr-trees. *The Int. Journ. of Robotics Research*, 35(11):1393–1416, 2016.

[7] Dirk V. Arnold and Nikolaus Hansen. Active Covariance Matrix Adaptation for the (1+1)-CMA-ES. In *Genetic And Evolutionary Computation Conference*, pages 385–392, Portland, United States, July 2010. doi: 10.1145/1830483.1830556. URL https://hal.archives-ouvertes.fr/hal-00503250.

[8] K. Schwab. *The Fourth Industrial Revolution*. World Economic Forum, 2016. ISBN 9781944835002.

[9] Ray Kurzweil. *The Singularity is near: When Humans Transcend Biology*. Viking, New York, 2005. ISBN 978-0-670-03384-3.

[10] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *IJRR*, 6(2):3–15, 1987.

[11] B. Siciliano and J.-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Int. Conf. Advanced Robotics*, pages 1211–1216, 1991.

[12] L. Saab, O.E. Ramos, F. Keith, N. Mansard, P. Soueres, and J.-Y. Fourquet. Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Trans. on Robotics*, 29:346–362, Jan 2013.

[13] A. Del Prete, F. Nori, G. Metta, and L. Natale. Prioritized motion-force control of constrained fully-actuated robots: Task space inverse dynamics. *Robotics and Autonomous Systems*, 63:150–157, Jan 2015.

[14] L. Sentis and O. Khatib. Synthesis of whole body behaviours through hierarchical control of behavioral primitives. *Int. Journal of Humanoid Robotics*, pages 505–518, 2005.

[15] C. Ott, A. Dietrich, and A. Albu-Schäffer. Prioritized multi-task compliance control of redundant manipulators. *Automatica*, 53:416 – 423, 2015.

[16] J. Salini, V. Padois, and P. Bidaud. Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions. In *ICRA*, pages 1283–1290, 2011.

[17] M. Liu, S. Hak, and V. Padois. Generalized projector for task priority transitions during hierarchical control. In *ICRA*, pages 768–773, 2015.

[18] R. Lober, V. Padois, and O. Sigaud. Variance modulated task prioritization in whole-body control. In *IROS*, pages 1–6, 2015.

[19] M. Liu, Y. Tan, and V. Padois. Generalized hierarchical control. *Autonomous Robots*, pages 1–15, 2015. doi: 10.1007/s10514-015-9436-1.

[20] S.-I. An and D. Lee. Prioritized inverse kinematics with multiple task definitions. *ICRA*, pages 1423–1430, 2015.

[21] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal. A unifying framework for robot control with redundant dofs. *Autonomous Robots*, 24:1–12, Jan 2008.

[22] G. Nava, F. Romano, F. Nori, and D. Pucci. Stability Analysis and Design of Momentum-based Controllers for Humanoid Robots. . In *IROS*, 2016.

[23] D. Pucci, F. Romano, S. Traversaro, and F. Nori. Highly dynamic balancing via force control. In *HUMANOIDS*, pages 141–141, 2016.

[24] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, Jan 2001.

[25] Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pages 491–496, 2016.

[26] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi. Learning soft task priorities for control of redundant robots. In *ICRA*, 2016.

[27] Niels Dehio, René Felix Reinhart, and Jochen J. Steil. Multiple task optimization with a mixture of controllers for motion generation. In *IROS*, 2015.

[28] Sehoon Ha and C.Karen Liu. Evolutionary optimization for parameterized whole-body dynamic motor skills. In *ICRA*, 2016.

[29] Guillaume Collange, Nathalie Delattre, Nikolaus Hansen, Isabelle Quinquis, and Marc Schoenauer. Multidisciplinary optimization in the design of future space launchers. In *Multidisciplinary Design Optimization in Computational Mechanics*, pages 459–468. Wiley-Blackwell, 2013.

[30] Dirk V Arnold and Nikolaus Hansen. A (1+1)-CMA-ES for constrained optimisation. In *GECCO*, pages 297–304, 2012.

[31] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi. Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization. In *HUMANOIDS*, 2016.

[32] V. Modugno, G.. Nava, D. Pucci, F. Nori, G. Oriolo, and S. Ivaldi. Safe trajectory optimization for whole-body motion of humanoids. In *Humanoids 2017, Submitted to IEEE/RAS International Conf. on Humanoid Robots , 2017*, 2017.

[33] D.E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 94(4): 303–309, 1972.

[34] H. Hanafusa, T. Yoshikawa, and Y. Nakamura. Analysis and control of articulated robot with redundancy. In *IFAC World Congress*, volume 4, pages 1927–1932, 1981.

[35] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *Proc. Int. Conf. Intell. Robots and Systems - IROS*, pages 323–329, 1998.

[36] P. Hsu, J. Mauser, and S. Sastry. Dynamic control of redundant manipulators. *Journal of Robotic systems*, 6(2):133–148, 1989.

[37] A. De Luca and G. Oriolo. Issues in acceleration resolution of robot redundancy. In *IFAC Sympos. on Robot Control*, pages 93–98, 1991.

[38] K. Senda. Quasioptimal control of space redundant manipulators. In *AIAA Guidance, Navig., and Control Conf.*, pages 1877–1885, 1999.

[39] J.M. Hollerbach and K. Suh. Redundancy resolution of manipulators through torque optimization. *IEEE Journal of Robotics and Automation*, 3(4):308–316, 1987.

[40] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. 3:43–53, Jan 1987.

[41] A. De Luca and G. Oriolo. Efficient dynamic resolution of robot redundancy. In *American Control Conference*, pages 93–98, 1990.

[42] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2616–2624. Curran Associates, Inc., 2013.

[43] Serena Ivaldi, Jan Babič, Michael Mistry, and Robin Murphy. Special issue on whole-body control of contacts and dynamics for humanoid robots. *Autonomous Robots*, 40(3):425–428, 2016.

[44] Fabrizio Flacco, Alessandro De Luca, and Oussama Khatib. Prioritized multi-task motion control of redundant robots under hard joint constraints. In *IROS*, pages 3970–3977, Jan 2012.

[45] Andrea Del Prete, Francesco Nori, Giorgio Metta, and Lorenzo Natale. Prioritized motion force control of constrained fully-actuated robots: atask space inverse dynamics. *Rob. and Auton. Systems*, 63:150 – 157, 2015.

[46] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The Int. Journ. of Robotics Research*, 33(7):1006–1028, 2014.

[47] J. Salini, V. Padois, and P. Bidaud. Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions. In *ICRA*, pages 1283–1290, 2011.

[48] Karim Bouyarmane and Abderrahmane Kheddar. Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations. In *IROS*, pages 4414–4419, 2011.

[49] Mingxing Liu, Yang Tan, and Vincent Padois. Generalized hierarchical control. *Autonomous Robots*, 40(1):17–31, Jan 2016.

[50] Joris Vaillant, Abderrahmane Kheddar, and et al. Multi-contact vertical ladder climbing with an hrp-2 humanoid. *Autonomous Robots*, 40(3):561–580, 2016.

[51] Chonhyon Park, Jia Pan, and Dinesh Manocha. Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments. In Lee McCluskey et al., editor, *ICAPS*. AAAI, 2012.

[52] Emmanuel Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *ACC*, pages 300–306, 2005.

[53] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IROS*, pages 4906–4913, 2012.

[54] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Int. Conf. on Machine Learning*, pages 1049–1056, 2009.

[55] Riad Akrour, Abbas Abdolmaleki, Hany Abdulsamad, and Gerhard Neumann. Model-free trajectory optimization for reinforcement learning. *CoRR*, abs/1606.09197, 2016.

[56] Peter Corke. *Robotics, Vision and Control - Fundamental Algorithms in MATLAB®*. Springer Tracts in Advanced Robotics. Springer, 2017.

[57] Eljaik J., A. Del Prete, S. Traversaro, M. Randazzo, and F. Nori. Whole body interface toolbox (wbi-t): A simulink wrapper for robot whole body control. In *ICRA 2014, Workshop on MATLAB/Simulink for Robotics, Education and Research, 2014*, 2014.

[58] S. Chiaverini, B. Siciliano, and O. Egeland. Redundancy resolution for the human-arm-like manipulator. *Robotics and Autonomous Systems*, 8(3):239–250, Jan 1991.

[59] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *J. Dyn. Sys., Meas., Control*, 108 (3):163–171, 1986.

[60] S. Ivaldi, M. Fumagalli, F. Nori, M. Baglietto, G. Metta, and G. Sandini. Approximate optimal control for reaching and trajectory planning in a humanoid robot. In *IROS*, pages 1290–1296, 2010.

[61] Sung-Hee Lee and Ambarish Goswami. A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Auton. Robots*, 33(4): 399–414, 2012.

[62] Adriano Macchietto, Victor Zordan, and Christian R. Shelton. Momentum control for balance. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 80:1–80:8, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4.

[63] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Resolved momentum control: humanoid motion planning based on the linear and angular momentum. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. IEEE, 2003. doi: 10.1109/iros.2003.1248880. URL https://doi.org/10.1109/iros.2003.1248880.

[64] Silvio Traversaro, Daniele Pucci, and Francesco Nori. A unified view of the equations of motion used for control design of humanoid robots. *On line*, 2017. URL https://traversaro.github.io/preprints/changebase.pdf.

[65] S. Ivaldi, O. Sigaud, B. Berret, and F. Nori. From humans to humanoids: the optimal control framework. *Paladyn Journal of Behavioral Robotics*, 3(2):75–91, 2012.

[66] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[67] P. T. De Boer, D.P. Kroese, S. Mannor, and R.Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134, 2002.

[68] C. Igel, T. Suttorp, and N. Hansen. A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In *GECCO*, 2006.

[69] Eric Brochu, Vlad M Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. eprint arXiv:1012.2599, arXiv.org, December 2010.

[70] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[71] H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86(1): 97+, 1964. ISSN 00219223. doi: 10.1115/1.3653121. URL http://dx.doi.org/10.1115/1.3653121.

[72] Jonas Mockus. On bayesian methods for seeking the extremum and their application. In *IFIP Congress*, pages 195–200, 1977.

[73] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13 (4):455–492, Dec 1998. ISSN 1573-2916. doi: 10.1023/A:1008306431147. URL https://doi.org/10.1023/A:1008306431147.

[74] Daniel James Lizotte. *Practical Bayesian Optimization*. PhD thesis, Edmonton, Alta., Canada, 2008. AAINR46365.

[75] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 1015–1022, 2010.

[76] Jacob Gardner, Matt Kusner, Kilian Q. Weinberger, John Cunningham, and Zhixiang Xu. Bayesian optimization with inequality constraints. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 937–945. JMLR Workshop and Conference Proceedings, 2014.

[77] Matthew D. Hoffman, Eric Brochu, and Nando de Freitas. Portfolio allocation for bayesian optimization. In *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 327–336, 2011.

[78] Lorenzo Natale, Francesco Nori, Giorgio Metta, Matteo Fumagalli, Serena Ivaldi, Ugo Pattacini, Marco Randazzo, Alexander Schmitz, and Giulio Sandini. *The iCub Platform: A Tool for Studying Intrinsically Motivated Learning*, pages 433–458. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[79] Matteo Fumagalli, Serena Ivaldi, Marco Randazzo, Lorenzo Natale, Giorgio Metta, Giulio Sandini, and Francesco Nori. Force feedback exploiting tactile and proximal force/torque sensing. *Aut. Robots*, 33:4:381–398, 2012.

[80] Francesco Romano and et al. A whole-body software abstraction layer for control design of free-floating mechanical systems. In *IEEE Int. Conf. on Robotic Computing (IRC)*, pages 148–155, 2017.

[81] Ugo Pattacini, Francesco Nori, Lorenzo Natale, Giorgio Metta, and Giulio Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *IROS*, pages 1668–1674, 2010.