Master Thesis in Control Systems Engineering

Robust Co-Design for Canonical Underactuated Systems

Master Candidate

Federico Girlanda

Student ID 2019289

Examiners

Prof. Augusto Ferrante

Prof. Pietro Falco

DEI, University of Padua

Supervisors

Dott. Shivesh Kumar

Dott. Lasse Maywald

RIC, DFKI GmbH

Final Exam 10.10.23

Academic Year 2022/2023

Abstract

Optimal behaviours of a system to perform a specific task can be achieved by exploiting the coupling between trajectory optimization, stabilization and design optimization. The main objective of this thesis work is to analyze a novel co-optimization approach, which aims to improve the optimization results applicability to real world systems. This methodology has shown interesting advantages for underactuated systems, which are systems that have fewer actuators than degrees of freedom and thus require to make use of the passive dynamics to compensate for their lack of control inputs. Two co-design algorithms, namely Robust Trajectory Control (RTC) and RTC with Design optimization (RTCD), have been concieved, implemented and evaluated. While the first method optimizes the trajectory behavior and the cost matrices of a stabilizing Time-varying Linear Quadratic Regulator (TVLQR) by fixing the model parameters, the second algorithm adds a further optimization layer where a design optimization is performed. Both aim to maximize the system's robustness, measured by a time-varying Lyapunov-based Region of Attraction (ROA) analysis. This analysis provides an intuitive representation of the controller's robustness to off-nominal states and can also result with a formal guarantee of stability for the entire stabilized trajectory.

The proposed algorithms have been tested on two different underactuated systems: the torque-limited simple pendulum and the cart-pole. The experiments demonstrate an increased volume of the stabilizable state-space region, indicating improved robustness. Extensive simulations of off-nominal initial conditions have further validated the results, and real system experiments have shown an improved insensitivity to torque disturbances.

Contents

List of Figures

List of Tables

List of Acronyms

RTC Robust Trajectory Control

RTCD RTC with Design optimization

ROA Region of Attraction

TVLQR Time-varying Linear Quadratic Regulator

EOM Equations of Motion

DIRTRAN Direct Transcription

LQR Linear Quadratic Regulator

SOS Sums of Squares

DDP Differential Dynamic Programming

FDDP Feasibility-driven DDP

OCP Optimal Control Problem

CSV Comma Separated Values

DOF Degrees Of Freedom

EOM Equations Of Motion

CMA-ES Covariance Matrix Adaption Evolution Strategy

ODEs Ordinary Differential Equations

1 Introduction

1.1 Motivation

Nowadays, the robotics research community shows a growing interest for legged robots, such as quadrupeds and humanoids, which can perform highly dynamical tasks. Boston Dynamics has recently presented to the world a new set of skills of it's humanoid robot Atlas [1], which is again pushing the limits of locomotion, sensing, and athleticism. Prominent companies such as Tesla have invested significantly in these kind of systems. The last year they started to develop Tesla Bot [2], their own general purpose, bi-pedal, autonomous humanoid robot. This interest is related to the industrial framework moving towards an increased human-robotic collaboration. Robots like Digit of Agility Robotics are not designed with the goal of mimicking humans, they have to be able to work in spaces alongside people [3]. This capability comes with the requirement of many of the same physical characteristics. The field of human-centered robotics has been also of inspiration for Apptronik's systems like Astra, which have shown the ability to operate in close proximity to humans [4].

1.1.1 Modeling Legged Systems

Generally, legged systems are underactuated which means that they are characterized by fewer control inputs than degrees of freedom. This is often due to the lack of a fixed base or to a set of joint limits. Underactuation causes the need to take into account passive dynamics and utilizing it to achieve stability and

efficient movement.
The term kinodynamic motion planning refers to a motion planning problem where the plant is subject to dynamic constraint, like underactuation, in addition to kinematic constraints like obstacle avoidance. As a consequence of this coupling, the control problems are more challenging and the results are often highly affected by the so-called *sim-to-real gap*. This last phenomena refers to the discrepancy between the performance achieved through simulation-based optimization and the actual behavior of the robot in the real world. Studying simple underactuated systems can provide useful insights into the challenges of controlling legged systems. Some common examples of underactuated systems are: torque-limited simple pendulum, acrobot/pendubot, cartpole, rimless wheel and hopping leg. To further enhance the effectiveness of the analysis, recent studies have introduced additional types of underactuated systems. For instance, in [5], a novel brachiating robot has been developed, expanding the potential deployment scenarios for humanoids and animaloids. Furthermore, there has been a recent research focus on benchmarking these systems [6, 7]. In particular, [6] introduces RealAIGym (Real Athletic Intelligence Gym), a platform dedicated to benchmarking dynamic behaviors on real robots. This platform establishes a valuable baseline for the application of dynamic control algorithms on real hardware.

1.1.2 Nature-inspired co-optimization

Every healthy individual can learn to accomplish a common task, such as running. However, just a few exceptions can achieve extraordinary performances. Usain Bolt's career is a notable example of this, as he is widely considered to be one of the greatest sprinters of all time. Bolt's record-breaking performances were not just the result of his natural athleticism and speed, but also of a careful combined development of his body and technique. A study [8] on his kinematics showed that his superior sprinting performance was due to an impressively long stride (aerial phase) and an intense acceleration phase. Noticeably, he had uneven strides, caused by one of his legs being slightly longer than the other, which results in a running gait, that is more effective than that of his competitors. Similarly, in robotics, a combined optimization of structural parameters and trajectory control is important to effectively accomplish the desired motion tasks. A design optimization process tunes the hardware parameters of a system to

ensure the described task with the best desired behaviour. Reaching the goal requires to search for a motion trajectory and to compute the control policy that permits the closed-loop trajectory following. The best trajectory is obtained via a trajectory optimization step, while the control input is computed by an appositely defined controller. In optimal control theory, the control signal comes from the formulation of the optimal cost-to-go, a function that returns the accumulated cost when running the optimal controller from every initial state to the goal. The power of holistic approaches has been demonstrated by Boston Dynamics' Atlas robot. It's ability to perform backflips, parkour, and dance with remarkable agility and naturalness is related to a combined optimization process [9]. Their methodology combines a trajectory optimization applied on an unconstrained model and the optimal cost-to-go parameter coming from the trajectory stabilization. The control input is obtained via a real-time solution of the constrained optimization, which considers the cost-to-go as it's objective function.

1.2 Related works

An interesting point of view for a control problem in the field of robotics locomotion comes from dividing it in three interdependent domains. Optimal performance can then be attributed to a shared optimality between trajectory optimization, stabilization and design optimization. The design optimization process seeks for the best set of system's model parameters with respect to the given cost function. The same objective is shared by a trajectory optimization problem, here the decision variables are related to the state and input trajectory. Finally, the stabilization of a trajectory is usually handled by a feedback control law that make the closed-loop dynamics of the system able to achieve the desired motion. A ROA analysis can be coupled to the last process in order to gain informations about the robustness of the computed control policy.
Maywald et. al. [10] analyzed a co-optimization between design parameters and stabilizing controller for an Acrobot system. Here, a time-invariant ROA estimation has been employed for introducing the concept of robustness in the objective function.

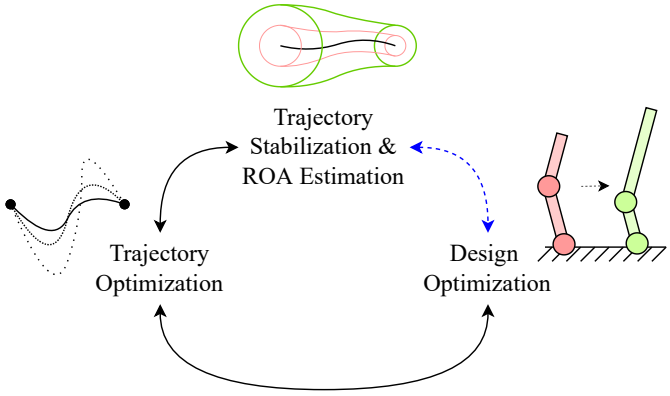

Figure 1.1: Interplay between optimization domains. [10]

Also the other interplays have been analyzed in the literature. A new version of a common trajectory optimization algorithm has been presented in [11]. The standard Direct Transcription (DIRTRAN) optimization problem finds an open-loop nominal state and input trajectory through a large, sparse nonlinear program. The new optimization problem, namely DIRTREL, incorporates linear feedback, bounded disturbances and a cost function that penalizes closed-loop deviations from the nominal trajectory. More recently, a bi-level optimization scheme has been introduced by [12] in order to compute an energy efficient design and trajectory for a jumping monoped. This work has also been expanded in [13], where a simulation-based robustness cost has been added to the optimization.

1.3 Contributions

The final purpose that inspires this thesis work is the future achievement of an athletic intelligence for legged systems. Our contribution consists in two algorithms, namely Robust Trajectory Control (RTC) and RTC with Design optimization (RTCD). The novelties contained in such methods are:

- Modular two layer optimization scheme for sampling-based co-design optimization.
- Time-varying ROA estimation (using SOS and Probabilistic methods) as a metric for co-design.

Two underactuated systems were considered in our verification process: the torque-limited simple pendulum and the cart-pole. The optimized task involves

the well-known "swing-up" control problem, it describes the problem of balancing an inverted pendulum by moving it from the hanging-down position to the upright position. This has been accomplished through the computation of a nominal trajectory via DIRTRAN, which has then been stabilized by a TVLQR controller.

1.4 Structure

In the next chapter a collection of the state of the art of co-design will be introduced. To a deep understanding of the selected tools and the implemented methodology, Chapter 3 recalls some fundamental mathematical backgrounds. In particular, details about system's Equations Of Motion (EOM) derivation, DIRTRAN trajectory optimization method, TVLQR control and ROA estimation are discussed. Thanks to these tools, the reader will be able to understand the outlined methods in Chapter 4. The proposed methodology will be then applied both in simulation and experimental verifications. Results are contained in Chapter 5. Finally, a general discussion is given in Chapter 6 in order to develop intuitions about further possible improvements.

2

State of the Art

This chapter mainly focuses on presenting existing research that inspired this thesis work. First, an overview of the last progresses on co-design in the field of robotics is introduced in Section 2.1. The use of ROA analysis in optimal control is then described in Section 2.2. Finally, in Sections 2.3 and 2.4, details about two recent results are discussed. These two implementation merge the previous concepts, deeply inspiring our work.

2.1 Co-Design in Robotics

A traditional approach to find the best trade-off between mechanical design and motion planning is to iterate between the two processes [14]. However, it is a challenging approach, especially for complex and dynamic robots like legged robots. Instead, concurrent design (co-design [15]) aims to automate this process by numerically optimizing both the motion and design parameters. Implementing a combined optimization has proven in the literature to provide better solutions [12, 16–21]. For instance, Ha et. al. [20] proposed a framework that had successfully optimized designs for legged robots performing tasks that include jumping, walking, and climbing up a step. Most recently, Fadini et al. [12] introduced a bi-level optimization scheme that modifies the actuator properties of a monoped in order to improve energy efficiency. Both are examples of the popular *sampling-based co-design*. This method are two-staged and exploit variants of Monte-Carlo sampling to find candidate robot designs. Each candidate is then evaluated through a motion planning stage. The Covariance Matrix

Adaption Evolution Strategy (CMA-ES) [22] is a commonly used sampling approach in this case. It uses a Gaussian prior on candidate design parameters and estimates a covariance matrix needed for the following sampling steps. A benefit of this approach is that it can use non smooth motion planners in the lower level. This means that it does not need informations about the gradient of the optimization cost. However, the algorithmic complexity of CMA-ES scales exponentially with respect to the number of design parameters (i.e., decision variables) due to the *curse of dimensionality* [23]. This limits its application to a reduced number of design parameters and constraints, which in turn limits its scalability, for instance to multiple tasks and environments. Other gradient-free optimization algorithms are available in the literature like the Nelder-Mead (NM) algorithm which has demonstrated its utility in the design optimization of parallel mechanisms [24]. On the other hand, a number of *gradient-based co-design* methods have been proposed in the literature. One approach is to formulate a single nonlinear program that optimizes both motion and design parameters. However, this can result in a non-modular method with a dangerously high algorithmic complexity. To avoid this problems derivative information have also been obtained via sensitivity analysis, but also in this case drawbacks can arise. In particular, the number of simultaneously optimized design parameters and the related constraints can be non-ideal. A solution has been proposed by Dinev et. al. [21] with a bilevel optimization approach tested on quadruped robots that jump and trot. It exploits the derivatives of the motion planning in the lower level into the higher level general-purpose nonlinear program that performs the design optimization.

2.2 Optimal Stabilizable Domains

One could argue that optimal control theories have provided a more comprehensive explanation of empirical phenomena in sensorimotor control than any other known class of methods. However, designs that perform well in theory may not translate to good performance in practice due to the so-called *sim-to-real* gap. This often holds for complex, non-linear and highly unstable systems like legged robots. Recent works in the literature are then trying to introduce the non-idealities in the optimization in order to minimize this difference [10, 11, 13, 25]. For instance, Manchester and Kuindersma [11] derived a tractable robust optimization algorithm, namely DIRTREL, that combines direct

transcription with linear-quadratic control design to reason about closed-loop reponses to disturbances. Gabriele Fadini [13] introduced a simulation-based cost in a bi-level co-optimization algorithm that aims to take into account a given set of disturbances. Recently, another method that has been studied to represent robustness is the ROA analysis [26], which defines a set of states that are guaranteed to be stabilized. A formal mathematical certificate of stabilizability can also be provided. This methodology has mostly been applied for stability analysis but it's use in space-filling algorithms like LQR-trees has obtained promising results [27–29].
ROA estimation has shown to be a powerful tool also in a co-design framework. In [10], the volume of the estimated region has been used as an optimization cost for the up-right stabilization of an Acrobot system. The resulting closed-loop system has then proven to be more robust to off-nominal initial state. A bi-level optimization scheme has been introduced by [13] in order to compute a robust energy efficient design and trajectory for a jumping monoped. A simulation-based robustness cost is considered for enhancing the result's robustness.
In the next sections, a more detailed explanation of some mentioned results have been outlined. For each related work, we will highlight the similarities to our approach and the improvements that we are proposing.

2.3 Robust Simulation Aided Co-Design

In [13] Gabriele Fadini and his research group expand their previous work [12] by adding a simulation-based robustness cost. Previously, they had formulated a bi-level optimization structure that minimizes energy consumption by tuning design and actuation parameters. Now, the optimization considers also the robust performances of the simulated system to noisy scenarios and actuator limits. The robustness metric comes from an evaluation of the controller performances over multiple disturbed simulations. Hence, it scales much better than other robust co-design approaches that explicitly optimize for a robust controller. The system is locally stabilized from the DDP trajectory optimization without considering the disturbances. A drawback of this kind of feedback is that it is guaranteed to work only close to the optimal trajectory. A possible solution that has been proposed is the introduction of a gain scaling in the feedback control law. This gain has then been considered as a further co-design variable.

Methodology

The algorithm is composed by two levels of optimization. The inner layer is characterized by a Feasibility-driven DDP (FDDP) implemented in Crocoddyl [30]. It improves the poor globalization strategy of Differential Dynamic Programming (DDP) and provides an optimal trajectory along with the related control commands for highly-dynamic maneuvers. This algorithm resembles direct multiple-shooting formulations with only equality constraints but it does not introduce extra decision variables, which often increases the computation time. The outer loop of the proposed bi-level structure is composed by a CMA-ES optimization of the design parameters. Here, the additional gain-scaling parameter is optionally added.

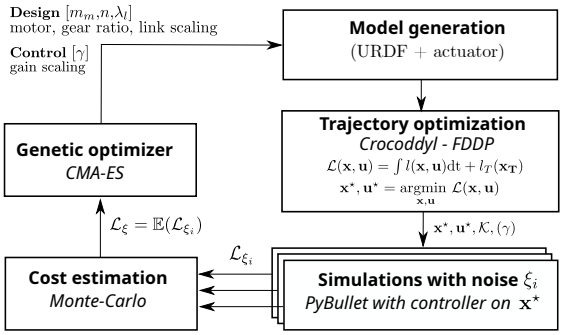

Figure 2.1: Robust bi-level scheme with simulations. [13]

Robust Cost

The CMA-ES is implemented to fix a new population of design parameters, and possibly of the control, at each iteration. An exhaustive introduction to this strategy will be provided in Section 4.2.2. The considered metric for the population selection is related to the ability of each stabilized trajectory to reject a given set of torque disturbances. Assuming a set of noises $\boldsymbol{\xi}_i \sim \mathcal{N}(0, \sigma^2)\mathbf{u}^*$, $i \in [0, N-1]$, which are modeled as a random variable with respect to the optimal control input $\mathbf{u}^*$. Given the optimal trajectory and input $(x^*, \mathbf{u}^*)$ from FDDP, the cost for each simulation is denoted as $\mathcal{L}_{\xi_i} = \mathcal{L}(x_{\xi_i}, \mathbf{u}_{\xi_i})$. $\mathcal{L}_{\xi_i}$, still being a random variable depending on the realisations of the noise $\boldsymbol{\xi}_i$, has an expected value $\mathcal{L}_\xi$ which can be obtained using a Monte-Carlo approach.

$$\mathcal{L}_\xi = \mathbb{E}(\mathcal{L}_{\xi_i}) \approx \frac{1}{N}\sum_{i=0}^{N-1} \mathcal{L}(x_{\xi_i}, \mathbf{u}_{\xi_i})$$

This is then the value that it is used to represent the robustness of each set of decision variables proposed by CMA-ES.

Optimization Strategy

Firstly, a population of possible robot hardware configurations is randomly initialized. Then, for each individual set of parameters, a model of the robot is generated and the corresponding inner Optimal Control Problem (OCP) solved. The optimal trajectory and control input performances are then evaluated within the robust cost computation procedure. Finally, this value is used for the population selection of the CMA-ES optimizer.

Results

Two different systems have been tested: a serial manipulator and a monoped. Both results in an improvement of the robustness in the stabilization of the required task. In the first case, a common *pick-and-place* task has been analyzed. Whereas the standard scenario was hardly applicable to the real system due to bandwidth limitations, the robust cost's improvement resulted in more stabilizable state trajectories. Hence, with better performances under real disturbances. The monoped faced instead a jumping task. The reference trajectories of the standard case were minimizing the cost, but such optimality did not translate to the real system, as these trajectories were not easy to follow in perturbed scenarios. After the optimization, a trajectory more near to the desired ones was obtained.
Overall, it has been noticed that the robustness increase coincide with a choice of a more transparent hardware. Transparent actuators can be obtained by minimizing friction and reflected inertias at the joint level, so with quasi-direct-drive actuation and low rotor inertia. This way the actuator bandwidth and back-drivability are both increased. These properties are necessary for proprioception and rapid control corrections.

2.3.1 Discussion and Inspiration

The obtained results show the effectiveness of the given structure and the possibility of the CMA-ES strategy to have good convergence properties and to avoid local minima in a gradient free optimization problem. Taking this work as

an inspiration, our outer design optimization loop will have a similar structure and we will exploit the same genetic optimizer. Since better robustness results were obtained by this paper by adding parameter in the controller policy, a different way of dealing with the inner optimization loop has been implemented in our work. We have choosen to optimize directly the controller parameters in order to better decouple the processes of trajectory optimization and stabilization with respect to the FDDP approach. This choice will then enhance the modularity of the final algorithm. Also, this allows the introduction of explicit constraints on the motion variables, which is not possible in FDDP. Here, the constraints are introduced as weighted costs in the optimization, which ends-up not ensuring the constraint compliance. Finally, different kind of representation has been selected in order to represent the robustness feature. Lyapunov-based ROA analysis has been used in our cost computation. In particular, the volume of this region as been used as a metric for the optimization.

2.4 Co-optimization of Acrobot Design and Controller

The work of Lasse Maywald and Felix Wiebe [10] proposes an approach to combine the optimization of design parameters and the stabilization via a Linear Quadratic Regulator (LQR) controller. They argue that this interplay has not been extensively studied in the literature. Furthermore, they approach this co-design process by introducing time invariant ROA estimation as an objective function. They also provide a meaningfull overview of the interplay between trajectory optimization/stabilization and design optimization (Figure 1.1).

Methodology

The test bench used to prove the effectiveness of the proposed method is an acrobot. This is an highly unstable underactuated system composed by an actuated joint and two rigid links connected by a passive joint. The considered task is the stabilization of the up-right position via LQR control.
The co-optimization goal is to find a physical design and controller parameters, such that the volume V_{ROA} of the ROA around the desired final state is maximized. For design optimization, the decision variables are m_2, the point mass at the tip of the second link, and the link lengths l_1 and l_2:

$$\begin{aligned} \max_{m_2, l_1, l_2} \quad & V_{ROA} \\ \text{subject to} \quad & m_{2,min} \leq m_2 \leq m_{2,max} \\ & l_{1,min} \leq l_1 \leq l_{1,max} \\ & l_{2,min} \leq l_2 \leq l_{2,max} \end{aligned}$$

Similarly, the optimization problem for finding the optimal control parameters is:

$$\begin{aligned} \max_{q_{ii}, r} \quad & V_{ROA} \\ \text{subject to} \quad & q_{ii,min} \leq q_{ii} \leq q_{ii,max} \quad \forall i \\ & r_{min} \leq r \leq r_{max} \end{aligned}$$

where q_{ii} and r are the (diagonal) elements of the cost matrices $\mathbf{Q}$ and $\mathbf{R}$ of the LQR controller, respectively.

Robust cost

Robustness has been quantified by the volume of the state-space region V_{ROA} resulted from a sample-based time-invariant ROA estimation. This estimate has been performed by reasoning about the Lyapunov function properties in a similar manner of the one proposed by E. Najafi [31]. The algorithm formulation has been discussed in the background chapter, Section 3.4.2.
This estimation is possible thanks to a useful method for directly sampling from a sublevel set of the Lyapunov function V. This is a function of the state $\mathbf{x}$ that has to be zero if calculated in the desired state $\mathbf{x}^*$ and positive otherwise. The valid and simple solution that they are considering is the optimal cost-to-go coming from the LQR synthesis $J^*(\mathbf{x})$.

$$V(\mathbf{x}) = J^*(\mathbf{x}) = \bar{\mathbf{x}}^T \mathbf{S} \bar{\mathbf{x}}$$

where $\mathbf{S}$ is the solution of the so-called Algebraic Riccati Equations (ARE) [3.9] and the resulting quadratic form depends on the state in error coordinates $\bar{\mathbf{x}} = \mathbf{x}^* - \mathbf{x}$. Limiting the V magnitude with a scalar value ρ will then define an hyper-ellipsoidal sublevel set. In practice, sampling from

$$V(\mathbf{x}) < \rho$$

can be implemented thanks to

$$\bar{\mathbf{x}} = (\sqrt{\mathbf{\Lambda}}\mathbf{W})^{-1}\mathbf{y}$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing the the eigenvalues of $\rho^{-1}\mathbf{S}$, $\mathbf{W}$ is a matrix of eigenvectors, and $\mathbf{y}$ is the state sampled directly from the unit ball.

Optimization strategy

Two different optimization strategies have been emploied in order to solve this problem either in an alternating or a combined fashion. For the alternating approach, they try to find the optimal design first (design first) and then seek for the optimal control parameters, or vice versa (controller first). In the combined approach, the entire optimization problem is solved at once. Two different gradient-free numerical optimization methods, Nelder-Mead and CMA-ES, have been compared in the alternating approach.

Results

The best result in terms of ROA improvement has been obtained with the "design first" approach that exploit the genetic optimization algorithm CMA-ES. The corresponding increase of the ROA volume can be noticed in Figure 2.2.

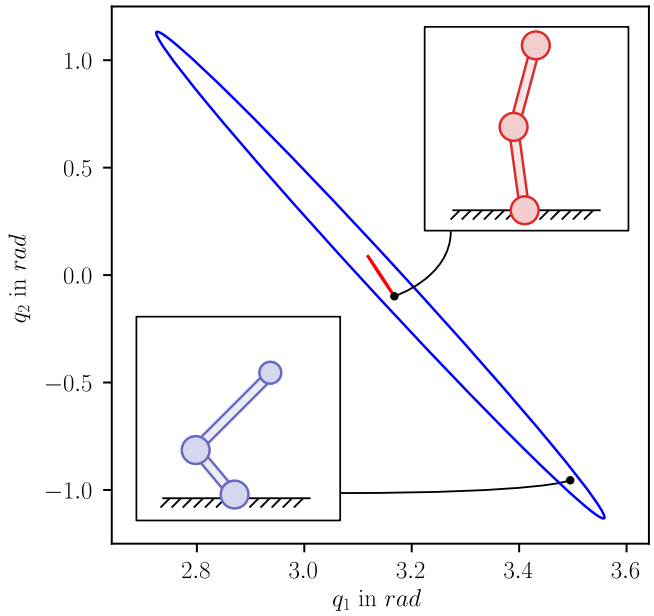

Figure 2.2: Acrobot ROA improvement.

The final ROA estimates have been verified by performing extensive simulations of the forward dynamics for initial conditions sampled randomly from the estimated ROA. No initial conditions were found, for which the top position could not be stabilized, which probabilistically verifies the ROA estimate.

2.4.1 Discussion and Inspiration

The results suggest an interesting possible approach to use optimization for improving the robustness provided by a stabilizing controller. In particular, it shows that the ROA volume can be used as a cost in an optimization problem for this purpose. As an extension, the time-varying version of the ROA analysis has been implemented in this thesis. We have considered a quadratic Lyapunov function coming from a linear quadratic regulator as well. Our approach is computationally heavier but it should be more powerful in terms of the final robustness. It makes also possible to deal with the whole swing-up control problem and not just the final stabilization. A ROA estimation method proposed in our work is the one based on Sums of Squares (SOS), which was not implemented in this paper. The obtained region will then come with a formal guarantee of stabilization. Furthermore, this paper shows again the power of the CMA-ES strategy in handling this kind of gradient-free optimization. As a further improvement, our verification step has also been integrated with real experiments, which provides a better intuition on the improved performances. Noticeably, the sampling method introduced in 2.4 has been fundamental for our implementation.

3

Mathematical Background

In this chapter a set of basic concepts for the overall thesis understanding is briefly presented. An overview of the modeling equations for mechanical systems is described in Section 3.1. The well-known methods that we have used to generate a trajectory and stabilize it are recalled in Sections 3.2.1 and 3.3.2. Finally, the ROA analysis has been adressed in more detail in Section 3.4.

3.1 Multibody Dynamics

The Equations Of Motion (EOM) are a set of equations that describes the time evolution of the behavior of a physical system in terms of its motion. A well known and wide spread procedure to obtain this mathematical viewpoint is the so-called Euler-Lagrange method [32]. This energy-based approach is particularly useful to understand the robot's dynamic properties and to analyze the control schemes. Using T as the total kinetic energy of the system, and U as the total potential energy of the system, $L = T - U$, and Q_i as the element of the generalized force vector corresponding to q_i, the Lagrangian dynamic equations are:

$$\frac{d}{dt}\frac{\delta L}{\delta \dot{q}_i} - \frac{\delta L}{\delta q_i} = Q_i \tag{3.1}$$

In the case of rigid robotic manipulators two particular conditions hold [33]:

1. The kinetic energy is a quadratic function of the vector $\dot{q}$ of the form

$$T = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}$$

where $\mathbf{q}$ is the n-dimensional joint position vector and the inertia matrix $\mathbf{M}(\mathbf{q})$ is a n × n positive definite matrix for each $\mathbf{q} \in \mathbb{R}^n$.

2. The potential energy $U = U(\mathbf{q})$ is independent of $\dot{\mathbf{q}}$.

Thanks to these properties we can write a general matrix form of Euler-Lagrange equations for robotic manipulators as:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \tau_g(\mathbf{q}) + \mathbf{B}\mathbf{u} \tag{3.2}$$

where $\mathbf{C}$ captures Coriolis forces, τ_g is the gravity vector and the matrix $\mathbf{B}$ maps control inputs into generalized forces. Note that $\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}}$ are paired together on the left side because these terms represents the force of inertia. Both Equations 3.1 and 3.2 can be applied for each specific robot configuration in order to obtain the related EOM.

3.2 Trajectory Optimization

The trajectory optimization problem deals with finding the optimal open-loop state and input trajectory for the accomplishment of a specific motion task, constrained by the system dynamics $f(\mathbf{x}, \mathbf{u})$. The obtained trajectory is also called *nominal trajectory* and it is often denoted by $(\mathbf{x}_0(t), \mathbf{u}_0(t))$. Given an initial condition, $\mathbf{x}_i$, and an input trajectory $\mathbf{u}(t)$ defined over a finite interval, $t \in [t_i, t_f]$, we can compute the long-term (finite-horizon) cost of executing that trajectory using the standard additive-cost optimal control objective and write the optimization problem [32] as:

$$\begin{aligned} \min_{\mathbf{u}(\cdot)} \quad & \ell_f(\mathbf{x}(t_f)) + \int_{t_i}^{t_f} \ell(\mathbf{x}(t), \mathbf{u}(t))dt \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad \forall t \in [t_i, t_f] \\ & \mathbf{x}(t_i) = \mathbf{x}_i \end{aligned} \tag{3.3}$$

This general formulation represents the dynamics integration by means of a generic integrator. Usually, additional constraints, such as collision avoidance or input limits, are also included. Constraints can be defined for specific time instants or for some subset of the trajectory. In order to formulate this as a numerical optimization, we must parameterize it with a finite set of decision variables. This parametrization can be obtained throught various methods which result in

different performances and characteristics. Direct methods discretize the problem directly, converting it into a constrained parameter optimization problem. This discretization is also referred as *transcription*. Building analitically necessary and sufficient conditions for optimality is needed for indirect methods before the discretization. This is often difficult but this methods still can be used in specialized applications where accuracy is critical.
Shooting methods reasons about forward simulation, avoiding to add $\mathbf{x}[\cdot]$ as a decision variable. Each state $\mathbf{x}[k]$ is defined by the dynamic evolution from the initial state $\mathbf{x}[0]$ with the input $\mathbf{u}[\cdot]$. For instance, for a discrete-time linear system it holds that

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k]$$

and hence

$$\mathbf{x}[k] = \mathbf{A}^k\mathbf{x}[0] + \sum_{j=0}^{k-1} \mathbf{A}^{k-1-j}\mathbf{B}\mathbf{u}[k]$$

However, this method presents few potential disadvantages with respect to transcription:

- Numerical conditioning. Shooting involves calculating $\mathbf{A}^k$ for potentially large k, which can lead to a large range of coefficient values in the constraints. This problem, also refferred as *tail wagging the dog*, is less important in the transcription approach.
- Adding state constraints. Having $\mathbf{x}[k]$ as explicit decision variable makes it easier.
- Parallelization. Whereas transcription permitts the evaluation of dynamic/constraints in parallel, shooting is more fundamentally a serial operator.

These differences becomes substantial for non-linear optimization problems.

3.2.1 Direct Transcription

Transcription of problem 3.3 and considering $\mathbf{x}[\cdot]$, $\mathbf{u}[\cdot]$ as decision variables leads to the so-called direct transcription. The time discretization is done for N points, the so-called *knot-points*.

$$\begin{aligned} \min_{\mathbf{x}[\cdot],\mathbf{u}[\cdot]} \quad & \ell_f(\mathbf{x}[N]) + \sum_{k=0}^{N-1} \ell(\mathbf{x}[k], \mathbf{u}[k]) \\ \text{subject to} \quad & \mathbf{x}[k+1] = g_d(\mathbf{x}[k], \mathbf{u}[k]), \quad \forall k \in [0, N-1] \\ & \mathbf{x}[0] = \mathbf{x}_i \\ & + additional\ constraints \end{aligned} \tag{3.4}$$

It can be noticed the use of a general function g_d for describing the discrete time evolution of the system's discrete dynamics f_d. The obtained sparse nonlinear problem permits to easily include state constraints and avoid numerical pitfalls such as the previously described *tail wagging the dog* effect, at the expense of a larger problem size [11]. This optimization problem can be solved using commercial sequential-quadratic programming (SQP) packages, such as SNOPT [34], that exploit the sparsity patterns in the linearized constraint matrix.
In practice, it is common to use a simple forward Euler integration scheme in g_d and to include the timestep as a decision variable. This last characteristic improves the integration performances and gives to the solver the freedom to choose the time length of the resulting trajectory. The new optimization problem can then be written as:

$$\begin{aligned} \min_{\mathbf{x}[\cdot],\mathbf{u}[\cdot],h} \quad & \ell_f(\mathbf{x}[N]) + \sum_{k=0}^{N-1} \ell(\mathbf{x}[k], \mathbf{u}[k]) \\ \text{subject to} \quad & \mathbf{x}[k+1] = \mathbf{x}[k] + f_d(\mathbf{x}[k], \mathbf{u}[k]) \cdot h, \quad \forall k \in [0, N-1] \\ & \mathbf{x}[0] = \mathbf{x}_i \\ & h_{min} \leq h \leq h_{max} \\ & + additional\ constraints \end{aligned} \tag{3.5}$$

It is worth pointing out that other integration schemes exist and can lead to a better result, e.g. midpoint integration. In fact, forwad Euler integration suffers of a, theoretically impossible, energy increasing if the chosen timestep is not small enough. This is because this integration scheme comes from the approximation of the derivative with the forward finite difference:

$$f_d(\mathbf{x}[k], \mathbf{u}[k]) \approx \frac{\mathbf{x}[k+1] - \mathbf{x}[k]}{h}$$

The approximation is valid as soon as $h \to 0$. A similar computation leads to backward Euler and midpoint methods.

3.3 Linear Quadratic Regulators

One of the most important and influential results in optimal control theory is the Linear Quadratic Regulator [32]. While solving a dynamic programming problem for continuous systems can be hard in general, there are few special cases where the solutions are very accessible. Most of these involve variants on the case of linear dynamics and quadratic cost. These are, in fact, suitable features for this kind of control policy.
Often the considered system won't be linear, a linearization near the state space region of interest will then be necessary. This procedure is usually done by considering the Taylor expansion of the nonlinear dynamics. For instance, the Taylor series of a real or complex-valued function $f(x)$ that is infinitely differentiable at a real or complex number a is the power series:

$$f(x) \approx \sum_{n=0}^{o} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

where o is the order of the approximation. The increase of o lead to a better approximation at the price of a slower computation.

3.3.1 Infinite Horizon Formulation

The local version of this regulator deals with the stabilization as $t \to \infty$ of a fixed-point, a desired final state and input. Firstly, a linearization of the dynamics around this point is performed, if necessary. The Taylor expansion of a generic nonlinear function f of two variables around a fixed point $(\mathbf{x}^*, \mathbf{u}^*)$ can be written as:

$$f(\mathbf{x},\mathbf{u}) \approx \underbrace{f(\mathbf{x}^*,\mathbf{u}^*)}_{:=0} + \sum_{n=1}^{o} \left(\frac{f_x^{(n)}(\mathbf{x}^*,\mathbf{u}^*)}{n!}(\mathbf{x}-\mathbf{x}^*)^n + \frac{f_u^{(n)}(\mathbf{x}^*,\mathbf{u}^*)}{n!}(\mathbf{u}-\mathbf{u}^*)^n \right) \tag{3.6}$$

Choosing a first order Taylor expansion, i.e. $o = 1$, and shifting the equations in error coordinates, $(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = (\mathbf{x} - \mathbf{x}^*, \mathbf{u} - \mathbf{u}^*)$, we can write:

$$\dot{\bar{\mathbf{x}}} = f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}^*, \mathbf{u}^*) \approx \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}}$$

It can be noticed that, $(\mathbf{x}^*, \mathbf{u}^*)$ is a fixed point of the dynamics f, hence

$$\dot{\bar{\mathbf{x}}} = f(\mathbf{x}, \mathbf{u}) \approx \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}} \tag{3.7}$$

Now, the shifted dynamics has a linear state-space form and it has a fixed point in the origin. Considering the infinite horizon cost function as

$$J = \int_0^\infty \left[\bar{\mathbf{x}}^T \mathbf{Q}\bar{\mathbf{x}} + \bar{\mathbf{u}}^T \mathbf{R}\bar{\mathbf{u}}\right] dt, \quad \mathbf{Q} = \mathbf{Q}^T \geq 0, \mathbf{R} = \mathbf{R}^T > 0$$

The goal is to find the optimal cost-to-go function which satisfies the Hamilton-Jacobi-Bellman (HJB) equation:

$$\forall \bar{\mathbf{x}}, \quad 0 = \min_{\bar{\mathbf{u}}} \left[\bar{\mathbf{x}}^T \mathbf{Q}\bar{\mathbf{x}} + \bar{\mathbf{u}}^T \mathbf{R}\bar{\mathbf{u}} + \frac{\delta J^*}{\delta \bar{\mathbf{x}}}(\mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}})\right]$$

In optimal control theory, it gives a necessary and sufficient condition for optimality of a control with respect to a loss function. It is well known that for this problem the optimal cost-to-go function is quadratic [32]. So, choosing the form

$$J^* = \bar{\mathbf{x}}^T \mathbf{S}\bar{\mathbf{x}}, \quad \mathbf{S} = \mathbf{S}^T > 0 \tag{3.8}$$

lead to obtain an optimal control policy of the type $\bar{\mathbf{u}} = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}\bar{\mathbf{x}} = -\mathbf{K}\bar{\mathbf{x}}$, where $\mathbf{S}$ is the solution of the so-called Algebraic Riccati Equations (ARE):

$$0 = \mathbf{SA} + \mathbf{A}^T\mathbf{S} - \mathbf{SBR}^{-1}\mathbf{B}^T\mathbf{S} + \mathbf{Q} \tag{3.9}$$

Shifting back to the initial reference frame, the dynamics control input has been determined as: $\mathbf{u} = \bar{\mathbf{u}} + \mathbf{u}^* = -\mathbf{K}\bar{\mathbf{x}} + \mathbf{u}^*$.

3.3.2 Finite Horizon Formulation

Similar reasoning can also be applied to the stabilization of a trajectory leading to a fixed point in a finite time span $t \in \left[t_i, t_f\right]$. The resulting optimal

control policy is now $\bar{\mathbf{u}} = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}(t)\bar{\mathbf{x}} = -\mathbf{K}(t)\bar{\mathbf{x}}$, where $\mathbf{S}(t)$ can be obtained by backward integration of the so-called Differential Riccati Equations (DRE):

$$-\dot{\mathbf{S}}(t) = \mathbf{S}(t)\mathbf{A} + \mathbf{A}^T\mathbf{S}(t) - \mathbf{S}(t)\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}(t) + \mathbf{Q}$$

where

$$\mathbf{S}(t) \succ 0, \ \forall t \quad and \quad \mathbf{S}(t_f) = \mathbf{Q}_f$$

This formulation holds even if the dynamics is time-varying, i.e. of the form:

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}$$

This is true also if we have time dependent cost matrices Q and R. A powerful application of the above result regards the stabilization of a nominal trajectory $(\mathbf{x}_0, \mathbf{u}_0)$ defined in a finite time interval $t \in \left[t_i, t_f\right]$ via *Time-Varying LQR (TVLQR)* [32]. Similarly as done in (3.6), a linearization could be necessary in the case of nonlinear dynamics. The procedure is very similar but now the linearization point $(\mathbf{x}^*, \mathbf{u}^*)$ can be a non-fixed point of the dynamics. Hence, the relationship described in (3.7) is not always true. In fact, now the linearization is time dependent and the coordinate system moves along the trajectory:

$$\dot{\bar{\mathbf{x}}} = f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u}_0) \approx \mathbf{A}(t)\bar{\mathbf{x}} + \mathbf{B}(t)\bar{\mathbf{u}}$$

where $(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = (\mathbf{x} - \mathbf{x}_0, \mathbf{u} - \mathbf{u}_0)$ represents the shift to error coordinates.
The previous derivation of the optimal control policy still holds. Hence, the stabilizing control input will take the following form:

$$\mathbf{u} = \mathbf{u}_0(t) - \mathbf{K}(t)(\mathbf{x} - \mathbf{x}_0(t))$$

In order to stabilize the trajectory we have to consider that stability is an infinite horizon concept, i.e. it has to consider the dynamics behaviour as $t \to \infty$. This detail can be adressed if $(\mathbf{x}(t_f), \mathbf{u}(t_f))$ is a fixed point of the dynamics by imposing $\mathbf{S}(t_f) = \mathbf{S}_\infty$, where $\mathbf{S}_\infty$ is the solution of (3.9). In practice, this mean to set the cost matrices in t_f as the ones of a stabilizing infinite-horizon LQR, i.e. $\mathbf{Q}(t_f) = \mathbf{Q}_f = \mathbf{Q}$ and $\mathbf{R}(t_f) = \mathbf{R}$.

3.4 Region of Attraction Estimation

The ROA estimation is an analysis that can be applied to the closed-loop dynamics of a system in order to analyze the control behaviour. In particular, it gives information about the robustness of the controller to off-nominal states. These anomalities are usually due to the so-called *sim-to-real gap*. In other words, this analysis defines a set of states that are guaranteed to be stabilized by the analyzed controller. Under certain conditions and for simple systems, it is possible to determine this region analytically through the solution of Ordinary Differential Equations (ODEs). A study conducted by [35] illustrates the application of this derivation method to the torque-limited simple pendulum. Most often, this region is impossible to determine analytically hence an estimation has to be considered.
For a locally attractive fixed point, $\mathbf{x}^*$, the region of attraction to $\mathbf{x}^*$ is the largest set $\mathcal{D} \subseteq \mathcal{X}$ for which:

$$\mathbf{x}(0) \in \mathcal{D} \Rightarrow lim_{t\to\infty}\mathbf{x}(t) = \mathbf{x}^*$$

Meaning that the dynamic evolution has to end-up, or stabilize, at $\mathbf{x}^*$, which is inside the region [32]. Lyapunov analysis can be used to estimate the ROA within the framework of the so-called Lyapunov-based methods. Here, a particular function V, called Lyapunov function, can give information about the closed-loop behaviour as $t \to \infty$, e.g. local and global asymptotical stability [32]:

$$V(O) = 0 \quad and \quad V(\mathbf{x}) > 0,\ \dot{V}(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in \mathcal{D}\backslash\{O\}$$

If the above conditions hold for the fixed point in the origin O, then it is asymptotically stable. Without loss of generality this condition can be applied to other fixed points by mean of a translation in the state-space. Sublevel sets $\{\mathbf{x}|V(\mathbf{x}) < \rho\}$, $\rho > 0$, of a Lyapunov function are then used as approximations of the region of attraction [32].
As an initial guess for the Lyapunov function the optimal cost-to-go introduced in 3.8 is often considered. It is a quadratic polynomial that has a zero in the considered fixed point, hence it is a valid choice by construction. As a consequence of this choice, the obtained ROA estimation will have an hyper-ellipsoidal shape. It is worth mentioning that more complex or more dynamics-related choices of

$V(\mathbf{x})$ lead to a better estimation of the real state-space set.
Setting the Lyapunov function as mentioned makes the estimation to be just a matter of finding a suitable ρ. Two meaningful cases can be distinguished:

- Time-invariant case, deals with the task of stabilizing a fixed point. A single ROA around the desired state has to be found, i.e. one ρ value.
- Time-varying case, cope with the stabilization of a nominal trajectory. A vector $\boldsymbol{\rho} \in \mathcal{R}_+^N$ has to be estimated, the resulting ROA is also called *funnel* [26].

Two methods to implement each of this cases will be presented. The first formulate the estimation problem via SOS optimization while the second evaluate the region around the nominal trajectory through sampling and simulations. Both are based on the results coming from Lyapunov theory.

3.4.1 Sums of Squares Optimization

For the class of systems with polynomial dynamics the estimation problem can be formulated as an optimization problem using sums-of-sqares (SOS) optimization [36]. This method has advantages in terms of scalability and does not require numerical simulations, that can be computationally expensive. On the other hand, it requires to express the closed-loop dynamics in a polynomial form. This might need the use of Taylor approximation, hence introducing differences with the real state evolution. Noticeably, the resulting invariant set is characterized by a mathematical certificate of stability, an inner approximation of the real ROA.
Sums-of-squares optimization effectively gives informations about the positivity of a polynomial $\forall \mathbf{x} \in \mathcal{X}$. If we just need to impose this condition on a limited region of the state-space $\mathcal{X}$, as in the Lyapunov condition, we can exploit the so-called S-procedure [37]. Given a polynomial $p(\mathbf{x})$ and a semi-algebraic set $g(\mathbf{x})$, where g is a vector of polynomials. If a polynomial "multiplier", λ, can be found such that

$$p(\mathbf{x}) + \lambda(\mathbf{x})^T g(\mathbf{x}) \quad is\ SOS \quad and \quad \lambda(\mathbf{x}) \quad is\ SOS$$

then this is sufficient to demonstrate that

$$p(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \{\mathbf{x} | g(\mathbf{x}) \leq 0\}$$

In this formulation λ has similar meaning as the well-known *Lagrangian multiplier*.

Time invariant estimation process

For the time-invariant case, with $\dot{\mathbf{x}} = f(\mathbf{x})$, we can verify stability by finding a sub-level set $\mathcal{D}$ limited by ρ of the Lyapunov function $V(\bar{\mathbf{x}})$, where $\bar{\mathbf{x}} = (\mathbf{x} - \mathbf{x}^*)$, such that the Lyapunov condition holds. $\mathcal{D}$ is then a verified region of attraction for the system, and any state within the region will converge asymptotically to $\mathbf{x}^*$ as $t \rightarrow \infty$. The optimization problem [32] can be formulated as:

$$\begin{aligned} \max_{\rho,\lambda} \quad & \rho \\ \text{subject to} \quad & -\dot{V} + \lambda(V - \rho) \quad is\ SOS \\ & \lambda \quad is\ SOS \\ & \rho > 0 \end{aligned} \tag{3.10}$$

where, thanks to the S-procedure, we are finding the maximum ρ while imposing that

$$\forall \mathbf{x} \in \{\mathbf{x} | V(\mathbf{x}) < \rho\},\ \rho > 0 \quad \Rrightarrow \quad \dot{V}(\mathbf{x}) < 0$$

which means that we are searching for the biggest sublevel set of the Lyapunov function V such that the Lyapunov condition for stability holds.
This formulation is not convex in ρ, so that it has to be solved with a bilinear alternation by fixing at each step the *Lagrangian multiplier* or ρ. However, since the problem is convex with ρ fixed and ρ is just a scalar, a simple line search can be performed in order to find the largest value for which the convex optimization returns a feasible solution. Another possible formulation is the so-called equality constrained formulation [38] which results in an optimization problem jointly convex in λ and in ρ, so that it can be solved in a single optimization step.

Time varying estimation process

Estimating the funnel around a nominal trajectory requires to solve a more involved optimization problem. Each knot point will have an associated estimated region, so $\boldsymbol{\rho} \in \mathcal{R}^{N_+}$ is no more scalar. While the definition of the goal ROA is the one above, the others have a slightly different meaning. Every state sampled from the knot point k must end up, following the system's dynamics,

inside the ellipse $k+1$. Hence, an estimation procedure for each couple of knot points is necessary. Each estimation process is now a bilinear alternation between two optimization problems:

1. Multiplier Step, searching for a feasible mutiplier by fixing $\boldsymbol{\rho}$.
2. Rho Step, searching for the maximum $\boldsymbol{\rho}$ by fixing the multiplier.

Given a meaningful initial guess of $\boldsymbol{\rho}$, this two steps has to be alternatively solved until a convergence condition. This condition can be, for instance, related to the overall magnitude of the $\boldsymbol{\rho}$ vector. After a time-invariant $\boldsymbol{\rho}_N$ have been determined, a bilinear alternation for each couple of knot points has to be solved going backwords for $k \in [0, N-1]$. The methodology can be summarized with the two following optimization problems:

1. Multiplier Step:

$$\begin{aligned} \max_{\gamma_i, \lambda_i} \quad & \gamma_i \\ \text{subject to} \quad & -(\dot{V} - \dot{\boldsymbol{\rho}}_i) + \lambda_i (V - \boldsymbol{\rho}_i) - \gamma_i \quad is\ SOS \\ & \lambda_i \quad is\ SOS \\ & \gamma_i > 0 \end{aligned} \tag{3.11}$$

2. Rho Step:

$$\begin{aligned} \max_{\boldsymbol{\rho}_i} \quad & \boldsymbol{\rho}_i \\ \text{subject to} \quad & -(\dot{V} - \dot{\boldsymbol{\rho}}_i) + \lambda_i (V - \boldsymbol{\rho}_i) \quad is\ SOS \\ & \lambda_i \quad is\ SOS \\ & \boldsymbol{\rho}_i > 0 \end{aligned}$$

As the definition is now different also the Lyapunov condition has been imposed differently. By means of the S-procedure, we are now constraining the Lyapunov function to change slower than the $\boldsymbol{\rho}$ value for all the states at the boundary of the ROA. This is sufficient for the dynamics to stay inside the funnel for the whole of its evolution.
Some interesting applications of the funnel estimation via SOS can be found in the literature. For instance, [27] uses funnels computation via SOS optimization

along with LQR-Trees, a "space-filling" algorithm. The state space is filled with regions in which the dynamics is guaranteed to be stabilized to some trajectory which will lead to the goal region. This approach has been applied to a post-stall perching problem and it has resulted, in real system experiments with a 95 percent perching success rate over 147 flights for a wide range of initial speeds.

Dealing with input saturation

An important implementation detail regards the input saturation effect that has often to be considered in a control task. The inclusion of this property in the SOS problem formulation requires the use of additional multipliers. A time invariant one-input example is presented in [28]. By formalizing the input saturation and introducing a set of multiplier as

$$\mathbf{u}_- \leq \mathbf{u} \leq \mathbf{u}_+ \quad and \quad \lambda_i \; for \; i \in [1,7]$$

The following modification of the Lyapunov condition is obtained:

$$\begin{array}{rcll} & & -\dot{V}_- + \lambda_1(V-\rho) + \lambda_2(\mathbf{u}_- - \mathbf{u}) & \\ -\dot{V} + \lambda(V-\rho) \quad is\ SOS & \Rightarrow & -\dot{V} + \lambda_3(V-\rho) + \lambda_4(\mathbf{u}-\mathbf{u}_-) + \lambda_5(\mathbf{u}_+ - \mathbf{u}) & is\ SOS \\ \lambda \quad is\ SOS & & -\dot{V}_+ + \lambda_6(V-\rho) + \lambda_7(\mathbf{u}-\mathbf{u}_+) & \\ & & \lambda_i\ ,\ i \in [1,7] & is\ SOS \end{array}$$

where $\dot{V}_- = \dot{V}|_{\mathbf{u}=\mathbf{u}_-}$ and $\dot{V}_+ = \dot{V}|_{\mathbf{u}=\mathbf{u}_-}$. The new Lagrange multipliers serve to restrict the verification to the regions of interest. While the middle constraint is always evaluated, the other two are considered just in case of input saturation. As a consequence of this approach, the needed number of SOS conditions grows exponentially with the number of inputs. This has to be considered in systems where there is a big number of saturated inputs. The same solution and considerations still holds and can be easily extended for the Lyapunov constraint in the time-varying case.

3.4.2 Simulation-based Approach

Another method to estimate the ROA is based on repeated sampling and simulations. Hence, it can be computationally expensive and less scalable with respect to the previous approach. On the other hand, no approximation of

the dynamics is needed for the simulations and the overall implementation has turned out to be simpler.
As the number of simulations grows, the estimation becomes more fine. Eventually, an outer approximation of the real ROA is obtained. No formal guarantee of stabilization is provided by this procedure.

Time invariant estimation process

The estimation procedure also reasons about the Lyapunov stability theory. The time-invariant version estimates the ROA with an ellipse around the goal. Here, random states are sampled inside an initially guessed region and the Lyapunov condition is checked. If the condition holds the sampled state is inside the ROA and hence the proposed region remains valid. If not, the estimated region is shrunk by using the value of the Lyapunov function computed in that sampled state. A concise and meaningful representation of this algorithm is given in [31].

Algorithm 1 Memoryless sampling method for estimating the DoA

Require: $V(x), \dot{V}(x), n_s$
  $Initialize\ \rho = \infty$
  for $i = 1 : n_s$ **do**
    $Pick\ a\ random\ state\ x_i\ within\ the\ state\ space$
    if $\dot{V}(x_i) \geq 0$ **and** $V(x_i) < \rho$ **then**
      $\rho = V(x_i)$
    end if
  end for
  return ρ

If the number of simulations n_s is big enough, this algorithm leads to find a set of initial states that is probabilistically guaranteed to succeed the stabilization task. Also, a version with memory of this algorithm exists. It considers both a lower and an upper bound for rho that updates after each simulation depending on similar conditions used in Algorithm 1.

Time varying estimation process

A time-varying implementation of this estimation method is described in [29], the reasoning comes from a general extension of the previous one. They propose this method in the context of an LQR-Trees algorithm with probabilistic

coverage of the state-space. The methodology is based on simulations and falsifications. Initially, a funnel hypothesis is proposed, then each trajectory knot point is verified through sampling and simulations. Given a desired trajectory $(\mathbf{x}^*, \mathbf{u}^*)$ defined for N knot points, the initial ROA guess of node n is set as an open ball with radius $\epsilon_n > 0$, centered at $\bar{\mathbf{x}}_n$:

$$\mathcal{B}(\bar{\mathbf{x}}_n, \epsilon_n) := \{\mathbf{x}_n : d(\bar{\mathbf{x}}_n) < \epsilon_n\}$$

where $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$ and d is some metric on $\mathbb{R}^n$. Assume that a stabilizable region $\mathcal{G}$ around the goal state is given. For each node a set of samples is obtained and simulated through the end of the nominal trajectory. A simulation is considered successful if the sampled initial condition has been stabilized to the goal region, $\mathbf{x}_N \in \mathcal{G}$. If so, the verified ROA remains valid. If the simulation fails the initial region guess and the following ones are shrunk accordingly to the following rule:

$$\epsilon_{k,new} = min(d(\bar{\mathbf{x}}_k), \epsilon_{k,old}),\ \forall k \in [n, N-1]$$

A natural choice for d, suggested from the time invariant approach, could be the Lyapunov function V.

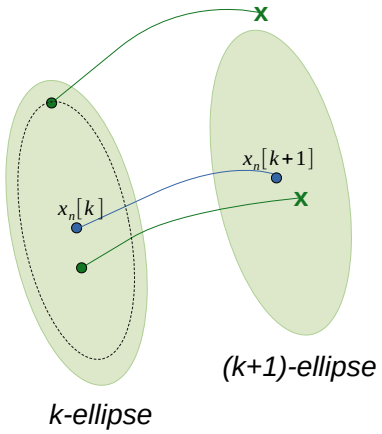

Figure 3.1: Simulation-based time-varying ROA estimation.

A variation of this formulation is also possible by considering an estimation procedure for each couple of knot points. The idea is to take inspiration from the SOS based approach presented in Section 3.4.1. The initial states for the simulation are sampled from the proposed ROA in k and they are simulated until the next time discretization $k + 1$. The same shrinking policy is then applied, a successful simulated trajectory is the one that ends up inside the ROA

in $k + 1$. In Figure 3.1 an intuitive representation of a successfull and a failing simulated trajectory has been proposed. It can be noticed how the shrinking process has been triggered by the failing simulation. The new k-ellipse will be now updated to become the dashed one.

4 Methodology

In this chapter, we will provide a comprehensive overview of the methodologies used in our co-design framework, building upon the tools and motivation introduced in Chapter 1 and the related works presented in Chapter 2. By providing a detailed description of our methodology, we aim to enable readers to better understand our approach, replicate our methods, and assess the validity and reliability of our results. The two proposed algorithms that form the core of our framework are introduced in Section 4.1. After outlining the optimization settings in Section 4.2, specific details that characterize our systems implementation are provided in Section 4.3.

4.1 Algorithms for a Robust Co-design

In this thesis, the analysis of the co-optimization between trajectory optimization, trajectory stabilization and design optimization has been adressed. Using as a starting point the instruments of the works presented in Chapter 2, we have tried to study how this different domains can be combined toghether.

Tool	**Objective**	**Input**	**Output**
DIRTRAN	Trajectory optimization	$M,\ Q,\ R$	$x_0(t),\ u_0(t)$
TVLQR	Trajectory stabilization	$M,\ Q,\ R,\ x_0(t),\ u_0(t)$	$K,\ S$
Funnel volume	Cost computation	$M,\ K,\ S,\ x_0(t),\ u_0(t)$	V

Table 4.1: Tools overview.

A functional input-optput description of the selected tools has been presented in

Table 4.1. The different processes are strictly related to each other, their requirements already define an intuitive structure for combining them. The DIRTRAN approach has been considered for the trajectory optimization step. In particular, the formulation presented in Equation 3.5 has been used. The reference trajectory $(\mathbf{x}_0(t),\ \mathbf{u}_0(t))$ has then been stabilized via the TVLQR controller that has been introduced in Section 3.3.2. As an implementation choice, the $\mathbf{Q}$ and $\mathbf{R}$ matrices of the stabilization have been also used for computing the quadratic cost that drives the search for a nominal trajectory:

$$\ell(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}\mathbf{u}(t)$$

Intuitively, these two processes are very close to each other and hence it is meaningful that they are aiming to minimize a similar cost function. On the other hand, this comes at the cost of computing a new trajectory for each proposed controller even if the design is fixed. This can result in a computationally expensive behaviour as the state dimension increases. For every stabilized trajectory, a funnel can be estimated by using one of the time-varying methods presented in Section 3.4. Finally, the volume of this ROA can be computed with the approximation that will be introduced in Chapter 4.2.1.

4.1.1 RTC

The simpler proposed algorithm is concurrently optimizing the nominal trajectory and the stabilizing controller, while fixing the design parameters $\mathbf{M}$. This has been implemented by considering $\mathbf{Q}$ and $\mathbf{R}$ as decision variables for the optimization process.
At each iteration the optimizer will propose a new set of cost matrices. Firstly, a new nominal trajectory is computed via DIRTRAN. The trajectory is then stabilized via TVLQR, which provides the tools to compute the related ROA. Finally, the volume of the estimated funnel is used by the CMA-ES strategy to weight each initially proposed set of costs. A detailed definition of this optimization strategy will be introduced in Section 4.2.2. It can be noticed that, thanks to the choice of using the same cost matrices for both trajectory optimization and stabilization, the solver has influence on both processes. A schematic view of the algorithm workflow is shown is Figure 4.1.

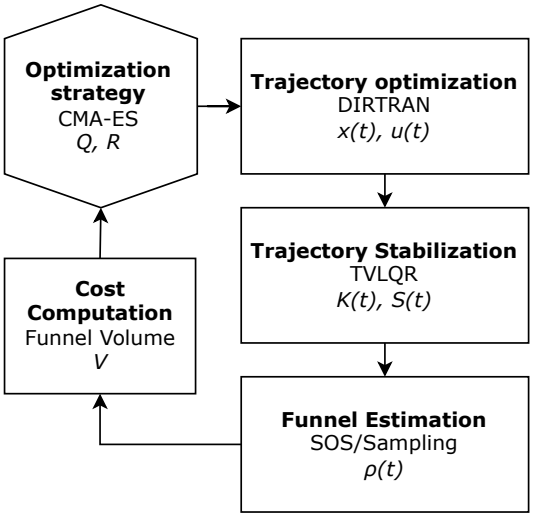

Figure 4.1: RTC algorithm scheme for a fixed design $\mathcal{M}$.

4.1.2 RTCD

To complete our analysis, a second algorithm with the capacity to vary the design parameters $\mathbf{M}$ has also been implemented. The resulting process is then more computationally expensive due to the increased number of optimization layers. As it is changing hardware parameters, it is also more difficult to verify in the real world. Even though the problem complexity increases, the solver has now more power to improve the objective function.

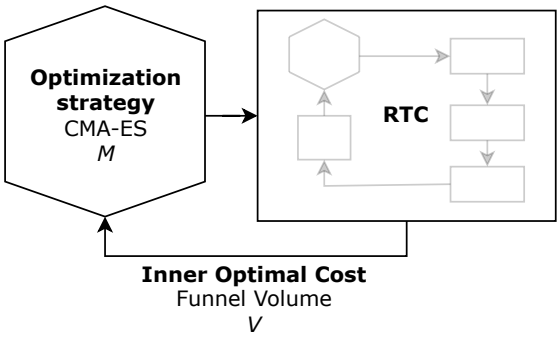

Figure 4.2: RTCD algorithm scheme.

An outer CMA-ES optimization layer has been added to the previous algorithm. The optimization strategy is now proposing a new set of model parameters for each iteration. The RTC optimization is then started by fixing $\mathbf{M}$. The inner-optimization will then come up with an optimal stabilized trajectory and

it's related funnel volume. Eventually, designs that minimize the optimal cost computed in the inner layer will be preferred by the solver.

4.2 Optimization Settings

In this section, we will outline the optimization settings used in our optimization algorithms. Specifically, we will present three key components of our optimization approach: cost function, strategy, and decision variables. The cost is a critical component of an optimization process as it defines the objective function that has to be optimized. It is designed to evaluate the performance of the proposed solution at each iteration. The optimization strategy is another fundamental component of an optimization approach. It defines the algorithmic techniques that are used to search for optimal solutions. A well-chosen optimization strategy can enable the optimization process to converge to optimal solutions quickly and efficiently. Finally, the decision variables represent the set of parameters that have to be optimized. These variables are typically selected based on their potential impact on the system's performance and the constraints that must be satisfied. It is essential to carefully define the optimization variables to ensure that they accurately capture the system's behavior and that they can be realistically optimized within the given constraints.

4.2.1 Cost function: The Funnel Volume

For evaluating the performance of our optimization algorithms, we have selected the volume of the funnel derived from time-varying ROA estimation as our metric. This region characterizes the off-nominal states that can still be stabilized by the computed control policy, and maximizing its dimension is our objective. As the funnel is a region in the state space, its volume depends on the state dimensionality. Our approach consists of a composition of N hyperellipsoids, where N is the number of knot points of the nominal trajectory. The volume of each hyperellipsoid V_k is estimated and then summed to obtain the total volume V of the funnel.

$$V = \sum_{k=1}^{N} V_k$$

It is well known, that a linear transform $\mathbf{T} \in \mathcal{R}^{m \times n}$ maps the unit ball in $\mathcal{R}^n$ to an ellipsoid in $\mathcal{R}^m$ [10]. Consider the n-ball $\mathcal{S}_n = \{\mathbf{y} | \mathbf{y}^T \mathbf{y} \leq 1\}$ and a transform $\mathbf{T}$ such that $\mathbf{y} = \mathbf{T}\mathbf{x}$. Expressing $\mathcal{S}_n$ in terms of $\mathbf{x}$ yields:

$$\mathcal{S}_n = \{\mathbf{x} | \mathbf{x}^T \mathbf{T}^T \mathbf{T} \mathbf{x} \leq 1\}$$

This describes an ellipse in $\mathbf{x}$, since $\mathbf{T}^T\mathbf{T}$ is symmetric and by definition positive definite. A diagonalization of $\mathbf{T}^T\mathbf{T}$ yields:

$$\mathbf{T}^T\mathbf{T} = \mathbf{W}^T \mathbf{\Lambda} \mathbf{W}$$

where $\mathbf{W}$ and $\mathbf{\Lambda}$ are matrices of eigenvectors and eigenvalues, respectively. Therefore, the transform $\mathbf{T}$ can be written as:

$$\mathbf{T} = \sqrt{\mathbf{\Lambda}}\mathbf{W}$$

Thanks to this formulation we can write the mathematical formula for the volume of each hyperellipsoid k as:

$$V_k = |det(\sqrt{\mathbf{\Lambda}}\mathbf{W})|^{-1} \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}$$

where Γ is the Euler's Gamma function.
It is worth noticing that the exactness of this measure is not critical, its primary purpose is to provide intuition to the optimization strategy regarding robustness. To estimate the volume at each knot point, we propose a sample-based method that leverages the concept of convex hull. Using $\mathbf{T}^T\mathbf{T} = \rho^{-1}\mathbf{S}$, the following transform allows us to sample randomly from a sublevel set of a quadratic Lyapunov function such as $V(\mathbf{x}) = \mathbf{x}^T\mathbf{S}\mathbf{x} \leq \rho$:

$$\bar{\mathbf{x}} = (\sqrt{\mathbf{\Lambda}}\mathbf{W})^{-1}\mathbf{y}$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing the the eigenvalues of $\rho^{-1}\mathbf{S}$, $\mathbf{W}$ is a matrix of eigenvectors, and $\mathbf{y}$ is the state sampled directly from the unit ball. Specifically, we have tuned this sampling method to obtain states lying on the boundary of each hyperellipsoid. We then consider the convex hull defined by these samples to approximate the desired quantity.

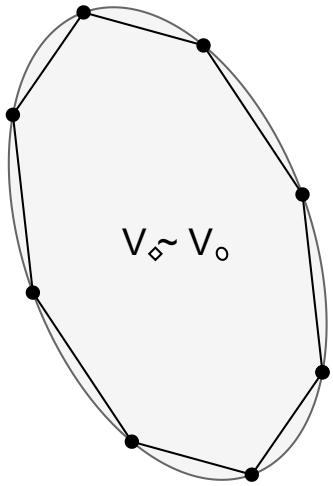

Figure 4.3: Estimating a state-space area via a convex hull approximation.

In Figure 4.3, we illustrate the concept of convex hull and how it can be used to estimate the area of a region in the state-space. The convex hull represents the smallest convex shape that encloses all the samples. This approach can be extended to higher dimensions, where the convex hull can represent the boundary of a complex region in the state-space. However, it is important to note that the sampling process becomes more challenging as the dimensionality of the state-space increases, due to the exponential increase in the number of samples required to achieve a certain level of accuracy. This problem is usually referred as *curse of dimensionality*

4.2.2 Optimization Strategy: CMA-ES

CMA-ES is a powerful stochastic optimization algorithm that is specifically designed to tackle non-convex and ill-conditioned objective functions. It is a gradient-free method that operates by maintaining a population of candidate solutions, also known as individuals. In each iteration of the optimization process, new candidate solutions are generated by varying the current population, with a preference for better-performing individuals. Inspired by principles from biological evolution, the CMA-ES algorithm adapts the covariance matrix of the search distribution to better match the landscape of the objective function. This approach allows it to effectively search complex, multi-modal search spaces with limited function evaluations.

Notably, in a comprehensive survey of black box optimization algorithms [39], Hansen et al. found that variants of the CMA-ES algorithm outperformed 31 other optimization algorithms in tackling difficult functions.
Overall, the CMA-ES algorithm has proven to be a highly effective tool for optimizing complex objective functions and has been successfully applied in a variety of fields, including machine learning, engineering, and finance.
For our analysis, we have considered the implementation of the CMA-ES algorithm from [22]. The main parameters that has been set for the optimization are grouped in Table 4.2.

Parameter	**Meaning**
X_0	Initial values for the decision variables.
σ_0	Initial standard deviation, the optimum is expected to lie within about $x_0 \pm 3\sigma_0$.
bounds	Boundaries for the decision variables.
maxfevals	Max number of evaluations for the cost function per each iteration.

Table 4.2: Main CMAES parameters.

In order to speed-up the optimization, the multiprocessing module *EvalParallel2* has been used. A simple modification in the code permits to evaluate the objective function in parallel.

4.2.3 Decision Variables: Stabilization and Design

In general, the proposed objective function highly depends on the closed-loop dynamics of the system. Hence, the costs matrices of the TVLQR stabilization and the design parameters have been considered as decision variables. Optimization problems with a large number of variables can quickly become computationally intractable, especially if the problem is non-linear or non-convex. By reducing the number of decision variables, the computational burden is reduced, and the optimization process becomes more efficient. Hence, choosing few but meaningfull optimization variables is fundamental. Some simplifications have then been taken into account. The **Q** and **R** matrices have, by definition, to be symmetric and respectively positive semi-definite and positive definite. A simple valid structure is then the diagonal one. Given a system with an n-dimensional state-space and m inputs, the **Q** and **R** matrix considered

parametrization is:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & q_{nn} \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} r_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & r_{mm} \end{bmatrix}$$

where $q_{11-nn} \geq 0$ and $r_{11-mm} > 0$.
It can be noticed how the input and state dimensionality increasing causes an increase of the number of decision variables. However, this parametrization can be further simplified by fixing some of the diagonal elements thanks to the knowledge about the specific systems dynamics and control. The same intuition-based simplification has been considered for the design parameters. Their contribution can be effective even if few parameters can vary. In general, hardware modifications can be time and resources expensive, hence minimizing them could be an implementation advantage.

4.3 Application: Swing-up and Stabilization

The swing-up stabilization problem is a common example problem for benchmarking and testing new approaches in the field of underactuated systems. It is a challenging task because it involves overcoming significant energy barriers to bring the system from a low-energy state to a high-energy one. This requires careful planning and control of the system's motion, as well as the ability to generate and manage energy in the system.
In this section, we will demonstrate the practical applications of our proposed optimization algorithms. Specifically, we will leverage the Drake toolbox to compute a TVLQR stabilization and to implement the SOS-based estimation of the ROA for underactuated systems. We will consider two testbenches, a torque-limited simple pendulum and a cart-pole, to highlight the effectiveness and limitations of our approach.

4.3.1 The DRAKE Toolbox

Drake is a C++ toolbox [40], initially created by the Robot Locomotion Group at the MIT Computer Science and Artificial Intelligence Lab (CSAIL) and now maintained by the Toyota Research Institute. This toolbox is designed to facili-

tate the analysis of robot dynamics and the construction of control systems, with a particular emphasis on optimization-based design and analysis.
While several simulation tools exist in the field of robotics, they tend to operate like black boxes, receiving commands and outputting sensor data. Drake, on the other hand, aims to simulate even the most complex robot dynamics, such as those involving friction, contact, and aerodynamics. Moreover, it is designed to expose the underlying structure of the governing equations, including sparsity, analytical gradients, polynomial structure, and uncertainty quantification. This approach makes it possible to employ advanced planning, control, and analysis algorithms. Drake also provides a Python interface to enable rapid prototyping of new algorithms and to offer open-source implementations for many state-of-the-art algorithms. Overall, the Drake toolbox provides a comprehensive and sophisticated set of tools for analyzing robot dynamics and building control systems, and has been widely adopted in both academic and industrial settings.
For the sake of this thesis work, the $FiniteHorizonLinearQuadraticRegulator(\cdot)$ and the $MathematicalProgram(\cdot)$ classes are the most important in the toolbox. The first one provides a straightforward method for managing TVLQR control. It solves the differential Riccati equation to compute the optimal controller and optimal cost-to-go for the finite-horizon linear quadratic regulator. This function handles system linearization by taking gradients of the provided continuous-time dynamics. The second one stores the decision variables, the constraints and costs of an optimization problem. It has been fundamental for the implementation of the SOS-based ROA estimation.

4.3.2 Torque-limited Simple Pendulum

The torque-limited simple pendulum system consists in an actuated rotational joint connected to a weight of mass m by a rigid link of length l. The total number of Degrees Of Freedom (DOF) is 1 and the related joint is actuated. The underactuation property here comes from the limit τ_{lim} on the torque τ that can be applied by the motor. During the motion the worst case scenario is when the weight is couteracting the lift with the maximum force: when the angle θ from the hanging position and the current one is 90 degrees. In this case the gravity force momentum is $\tau_g = mgl$ and the input torque has to apply a bigger torque

to complete the lift. This robot is then considered underactuated iff:

$$\tau \in \{-mgl < \tau < mgl\}$$

This rather simple system has been extensively used in the literature to test and benchmark different control approaches. A schematic representation of it can be seen in Figure 4.4.

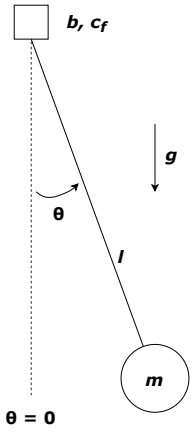

Figure 4.4: Schematic representation of the pendulum.

Equations of Motion

Equation 3.1 can be applied to obtain the EOM for this system. Consider $q = \theta$ and let Q model a damping torque with and an input torque τ, then we can write:

$$\frac{d}{dt}\frac{\delta L}{\delta \dot{\theta}} - \frac{\delta L}{\delta \theta} = \tau - b\dot{\theta}$$

where b is the damping factor.
From basic physics reasonings we know that the kinetic energy is $T = \frac{1}{2}ml^2\dot{\theta}^2$ and the potential energy is $U = mgl(1 - cos(\theta))$. Hence, the total mechanical energy L is

$$L = T - U = \frac{1}{2}ml^2\dot{\theta}^2 - mgl(1 - cos(\theta))$$

Introducing this last result in the previous equation and taking some mathematical simplifications, the dynamics equation can be written as:

$$ml^2\ddot{\theta} = \tau - mglsin(\theta) - b\dot{\theta}$$

Considering the 2 dimensional state $\mathbf{x} = \left[q, \dot{q}\right]^T$, the EOM can be written in state-space representation as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta} \\ \frac{\tau - mglsin(\theta) - b\dot{\theta}}{ml^2} \end{bmatrix}$$

Noticeably, the Coulomb friction c_f has been neglected.

Cost function

For this system we were able to implement both time invariant and time-varying ROA estimation via SOS, as described in detail in Section 3.4.1. The resulting inner estimation of the real invariant domain comes along with a stabilizability guarantee. Furthermore, the limited dimension of the state-space leads to a computationally tractable estimation. This is why we have chosen this estimation method over the simulation based one. The low dimensional state also allows an accurate volume calculation with the convex hull formulation, even with a relatively small number of sample points.

Decision Variables

With $\mathbf{x} \in X = \mathbb{R}^2$ and $u \in U = \mathbb{R}^1$, we can parametrize the TVLQR cost matrices as discussed in Section 4.2.3:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 \\ 0 & q_{22} \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} r_{11} \end{bmatrix}$$

with $q_{11} \geq 0$, $q_{22} \geq 0$ and $r_{11} > 0$.
The design parameters that are more easy to modify and meaningful for the cost function are the load mass and the link length. Both have been considered in the design optimization layer as:

$$\mathbf{M} = [m, l]$$

An overview of the full and reduced parameters set is presented in Table 4.3. Only mass and length are meaningful design parameters for the Pendulum system so no reduction has been introduced in this case. On the other hand, a decreasing of the number of decision variables has been imposed for the motion

optimization.

Design Opt	M_{full}	$[m,\ l]$
	$M_{reduced}$	$[m,\ l]$
Motion Opt	Q_{full}	$[q_{11},\ q_{12},\ q_{21},\ q_{22}]$
	R_{full}	$[r_{11}]$
	$Q_{reduced}$	$[q_{11},\ q_{22}]$
	$R_{reduced}$	$[r_{11}]$

Table 4.3: Decision variables overview for the Pendulum system.

4.3.3 Cart-Pole

The Cart-Pole is an underactuated robot composed by a cart, an actuated prismatic joint, with a pole mounted on it, by means of a passive rotational joint. The underactuation is due to the number of DOF (2) that is higher then the number of actuated joints (1).

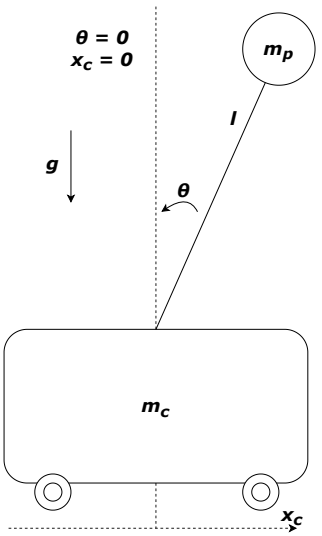

Figure 4.5: Schematic representation of the cart-pole.

This system can be considered as a simplified model to study complex systems like legged robots. In fact, It can be used to simulate the balancing and stepping motion of a leg during walking or running tasks.

Equations of Motion

Now consider $\mathbf{q} = [x_c, \theta]^T$ and $\mathbf{x} = \left[q, \dot{q}\right]^T$. The task is to stabilize the unstable fixed point $\mathbf{x}_f = [0, \pi, 0, 0]^T$, starting from the hanging straight down state $\mathbf{x}_i = [0, 0, 0, 0]^T$. The energy is given by

$$T = \frac{1}{2}(m_c + m_p)\dot{x}_c^2 + m_p\dot{x}_c\dot{\theta}lcos\theta + \frac{1}{2}m_p l^2\dot{\theta}^2$$
$$U = -m_p glcos\theta$$

Hence, the Lagrangian yelds the equations of motion:

$$(m_c + m_p)\ddot{x}_c + m_p l\ddot{\theta}cos\theta - m_p l\dot{\theta}^2 sin\theta = F_c - b_c\dot{x}_c$$
$$m_p l\ddot{x}_c cos\theta + m_p l^2\ddot{\theta} + m_p glsin\theta = -b_p\dot{\theta}$$

where F_c is the control input and it is the force applied to the cart, b_p and b_c are the damping parameters associated with the two DOFs. The nonlinear Coulomb friction applied to the linear cart, and the force on the linear cart due to the pendulum's action have been neglected.
For the sake of our implementation, a translation in the state space has been implemented for the angle θ: $\theta = \theta + \pi$. As a consequence, the task becomes to stabilize the point $\mathbf{x}_f = [0, 0, 0, 0]^T$, starting from the state $\mathbf{x}_i = [0, \pi, 0, 0]^T$. Furthermore, in our experiments we have considered a more detailed set of equations that were given in the documentation of the real system [41]:

$$(J_{eq} + m_p)\ddot{x}_c + m_p l\ddot{\theta}cos\theta - m_p l\dot{\theta}^2 sin\theta = F_c - B_{eq}\dot{x}_c$$
$$m_p lcos\theta\ddot{x}_c + (J_p + m_p l^2)\ddot{\theta} + m_p glsin\theta = -B_p\dot{\theta}$$

where J_* and B_* parameters deals with the modeling of the inertia and damping components.

Cost function

In this case, the time-varying ROA estimation has been obtained via the variation of the simulation-based approach described in Section 3.4.2. The time-varying SOS based implementation turned out to be more involved. The increased dimension of the state space caused a problem infeasibility that end-up preventing the estimation to be meaningful. However, the time invariant

version of this last method has been used to estimate the hyper-ellipsoid around the goal state. Intuitively, having an inner estimation of the last region is always a good starting point for a time-varying estimation.
The increase of the state dimensionality has affected, as expected, the computational time. The simulations were more time consuming and more samples were needed for accuracy of both funnel estimation and volume computation. The implemented estimation method is not providing formal guarantees. For the sake of an optimization procedure, is sufficient that the cost gives a quantitative information about the desired property. A trade-off between accuracy and timings has then been considered in our experiments.

Decision Variables

Since now the state $\mathbf{x} \in X = \mathbb{R}^4$, the parametrized stabilization cost matrices for this system are the following ones:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 & 0 & 0 \\ 0 & q_{22} & 0 & 0 \\ 0 & 0 & q_{33} & 0 \\ 0 & 0 & 0 & q_{44} \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} r_{11} \end{bmatrix}$$

Here, a further simplification has been introduced to lower the number of considered decision variables. The state weights q_{33} and q_{44} have been fixed to 100, a value that in practice was working well for our real design parameters.
As before, not all the design parameters has been considered but only:

$$\mathbf{M} = \begin{bmatrix} m_p, l \end{bmatrix}$$

For our system, these were the most straightforward to modify. Their effect on the system dynamics is also more intuitive thanks to the similarity with the previous system. An overview of the full and reduced parameters set is presented in Table 4.4. The mass of the cart has been fixed by assumption. Furtermore, the **Q** matrix has been simplified as depicted before. This last approximation lead to a considerable decrease of the number of decision variables and hence of the problem complexity.

Design Opt	M_{full}	$[m_p,\ m_c,\ l]$
	$M_{reduced}$	$[m_p,\ l]$
Motion Opt	Q_{full}	$[q_{11},\ q_{12},\ \ldots,\ q_{43},\ q_{44}]$
	R_{full}	$[r_{11}]$
	$Q_{reduced}$	$[q_{11},\ q_{22}]$
	$R_{reduced}$	$[r_{11}]$

Table 4.4: Decision variables overview for the Cart-pole system.

5 Results

This chapter contains the implementation of the proposed methodologies. The testbenches that we have used for our verifications are two underactuated robots: the torque-limited simple pendulum (Section 5.1) and the cart-pole (Section 5.2). For each system, the resulting optimization parameters have been firstly applied in simulation. Then, real world experiments have been implemented accordingly to get a stronger verification.

5.1 Torque-limited Simple Pendulum

The verification of the proposed algorithms has been first implemented on a torque-limited simple pendulum system.

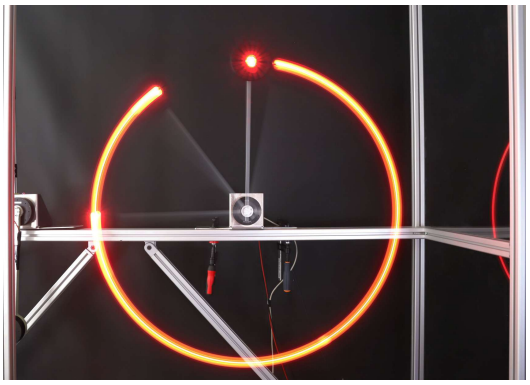

Figure 5.1: Real pendulum system.

The considered design parameters for both simulation and optimization have

been inspired by the real system which is availble in the *Underactuated Lab* of the DFKI GmbH Robotics Innovation Center in Bremen [42]. Here, an AK80-6 actuator from T-Motor is rigidly connected to a weight through an alluminium rod. A representation of this system achieving a swing-up task is shown in Figure 5.1.

5.1.1 Optimization Results

Two cases have been distinguished for this system, depending on the pendulum design parameters that has been considered. The first case has been used to get intuition about the optimization properties, while the second is suitable for our real system application.
Coulomb friction has not been taken into account for two reasons. First, the simulator from the *DRAKE* toolbox was initially used and it cannot handle this kind of parameter. Second, we argue that the effect of this parameter can be always avoided through friction compensation.

Optimization Properties Analysis

For the first optimization case, the considered initial design parameters are:

Parameter	**Value**
m	$0.67\ kg$
l	$0.5\ m$
b	$0.4\ Nm/s$
τ_{lim}	$2.5\ Nm$

Table 5.1: Initial Pendulum model parameters.

The system that handled the optimization computations is a 2-core 4-thread 2.70 GHz Intel(R) Core(TM) i7-7500U computer with 8 Gb of RAM. A maximum of 3 parallelized computations have been considered.
Both RTC and RTCD methods have been tested. The optimization settings and results are summarized in Table 5.2 and 5.3. The settings X_0, σ_0, $bounds$, $maxfevals$ refers to the CMA-ES parameters presented in Section 4.2.2. It can be noticed that both the optimization processes result in an improvement of the ROA volume.

CMA-ES Init	X_0	$[\, q_{11},\, q_{22}, r\,] = [10, 1, 0.1]$
	σ_0	0.1
	$bounds$ (upper)	$[10, 10, 1]$
	$bounds$ (lower)	$[1, 1, 0.1]$
	$maxfevals$	100
	fixed design	$[\, m,\, l] = [0.67, 0.5]$
	volume	15.75
RTC Opt	parameters	$[\, q_{11},\, q_{22}, r\,] = [9.9, 1.21, 0.95]$
	timing	23 min
	volume	40.8

Table 5.2: RTC optimization settings and results: Pendulum

CMA-ES Init	X_0	$[\, m,\, l] = [0.67, 0.5]$
	σ_0	0.1
	$bounds$ (upper)	$[0.9, 0.7]$
	$bounds$ (lower)	$[0.5, 0.3]$
	$maxfevals$ (outer)	20
	$maxfevals$ (inner)	20
	volume	15.75
RTCD Opt	parameters	$[\, m,\, l,\, q_{11},\, q_{22}, r\,] = [0.5, 0.3, 9.96, 1, 0.9]$
	timing	86 min
	volume	119.64

Table 5.3: RTCD optimization settings and results: Pendulum

The cost evolution is shown in Figure 5.3(left). More evaluations have been computed in the RTCD case, hence the execution time has increased. Each step represents an optimal solution obtained in an iteration of the solver computation. Every iteration needs maximum $maxevals = 20$ function evaluations. In the bi-level optimization case a maximum of $maxevals(inner) \times maxeval(outer) = 400$ function evaluations were available to the solver.
The final result of RTCD is way more robust than the one of RTC. We have noticed that the design optimization layer brings the system to violate the uderactuation condition $mgl > \tau_{lim}$. A further particular behaviour can be noticed in the r parameter of the RTC optimization, which is the one related to the control input. It's increasing is related to the improvement of the funnel volume. These behaviours are depicted in Figure 5.3(right).

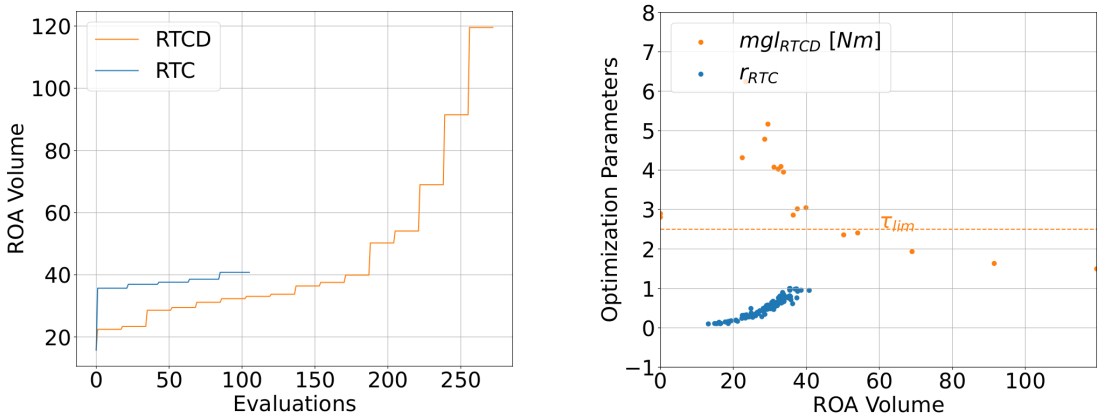

Figure 5.2: Ideal optimization evolution.

A comparison between initial and optimized scenario is proposed in Figures 5.3 and 5.4. The CMA-ES initial parameters will be from now on referred as DIRTRAN parameters. The funnel improvement is visible in both cases. In the RTC case we can see that the nomininal trajectory remains almost the same, but still the controller optimization ends-up improving the robustness.

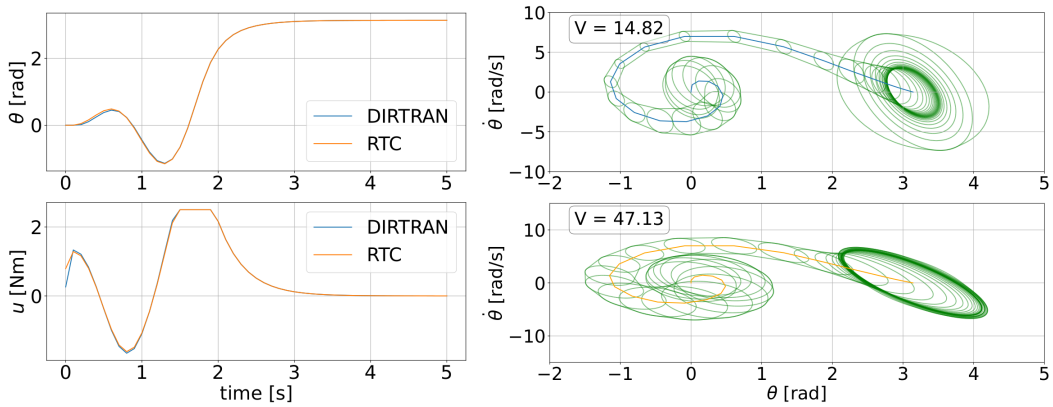

Figure 5.3: RTC vs DIRTRAN: nominal trajectories (left) and funnels (right).

In the RTCD optimized framework the trajectory changes due to the pendulum parameters optimization. The resulting funnel is way more big and the swing-up is achieved with only one pendulum swing. This last behaviour is typical of the fully-actuated situation. In fact, now we have obtained:

$$m_{RTCD} g l_{RTCD} \approx 1.48\ Nm < \tau_{lim}$$

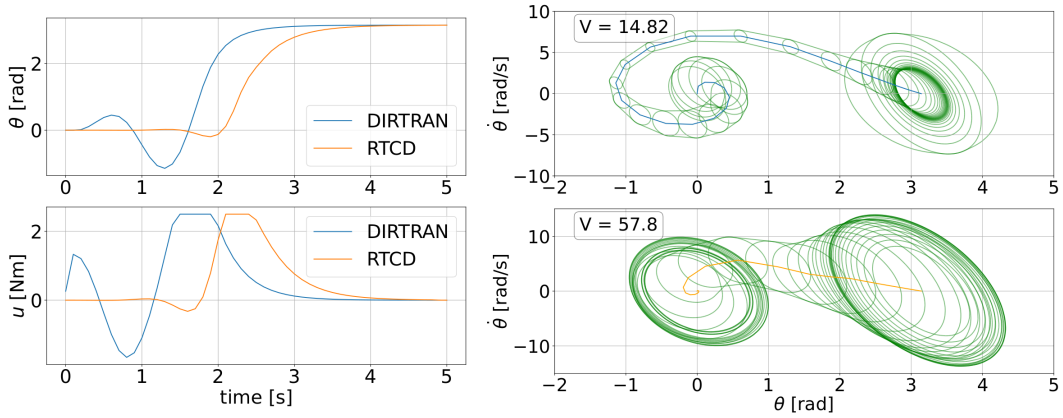

Figure 5.4: RTCD vs DIRTRAN: nominal trajectories (left) and funnels (right).

Real System Optimization

In the second optimization case, we have considered the real system hardware while fixing the optimization parameters. The initial design parameters have been set as:

Parameter	Value
m	$0.7\ kg$
l	$0.4\ m$
b	$0.1\ Nm/s$
τ_{lim}	$2.5\ Nm$

Table 5.4: Initial Pendulum real model parameters.

The available system mass is $m_{real} \approx 0.6\ kg$. By setting this value as a lower bound for the design optimization of the mass, we are expecting the optimization to choose it. This is because of the intuition gained from the first optimization case. It is easier for us to add weight rather then reduce it. Similarly, the real length is $l_{real} = l$, we will set it as a lower bound. Using the same rod for both initial and final design will reduce the timings for a real test. The optimization settings that are shown in Tables 5.5 and 5.6.

CMA-ES Init	X_0	$[q_{11}, q_{22}, r] = [10, 1, 0.1]$
	σ_0	0.9
	$bounds$ (high)	$[10, 10, 10]$
	$bounds$ (low)	$[1, 1, 0.001]$
	$maxfevals$	100
	fixed design	$[m, l] = [0.7, 0.4]$
	volume	14.83
RTC Opt	parameters	$[q_{11}, q_{22}, r] = [9.5, 1.2, 1.64]$
	timing	24 min
	volume	47.2

Table 5.5: RTC optimization settings and results: Pendulum

CMA-ES Init	X_0	$[m, l] = [0.7, 0.4]$
	σ_0	0.1
	$bounds$ (high)	$[0.8, 0.6]$
	$bounds$ (low)	$[0.6, 0.4]$
	$maxfevals$ (outer)	20
	$maxfevals$ (inner)	40
	volume	14.83
RTCD Opt	parameters	$[m, l, q_{11}, q_{22}, r] = [0.61, 0.4, 9.98, 1, 3.387]$
	timing	3 h
	volume	57.84

Table 5.6: RTCD optimization settings and results: Pendulum

The computations have been handled by a CPU-cluster available in the DFKI research institute. It consists in 12 compute nodes, each having 2x 8-core Xeon E5-2630 v3 (Haswell), 128GB ECC RAM. Still a maximum of 3 parallelized processes has been implemented for consistency with the last results. However, multiple nodes have been used to speed-up the algorithms testing phases.
The optimized volume has increased almost 4 times the initial value for both RTC and RTCD. The cost evolution during the optimization can be seen in Figure 5.5(left). The expected parameters behaviour has been achieved from the design optimization. The similar parameters behaviour can be noticed in Figure 5.5(right).

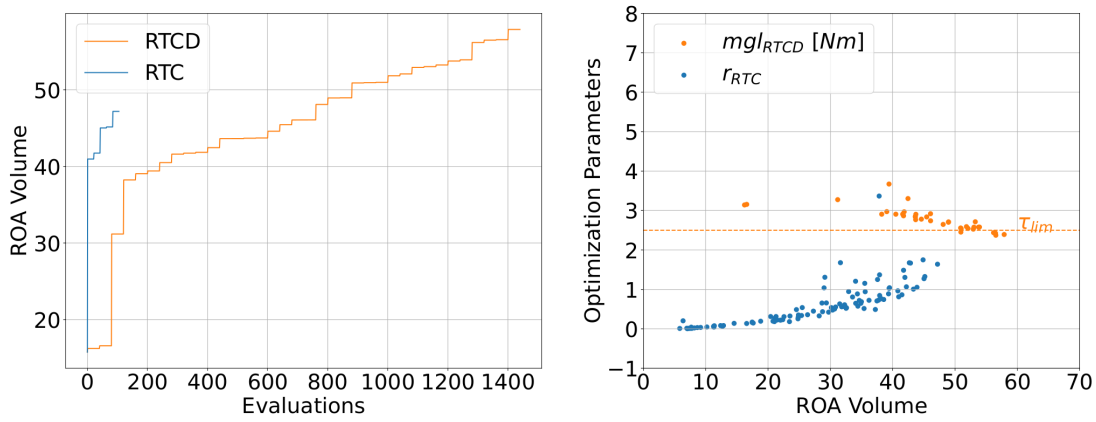

Figure 5.5: Optimization results for simple pendulum.

The increased shape of the funnel is depicted in Figure 5.6. Here, the result of the RTC optimization is compared with the initial situation. Even though, as expected, the two nominal trajectories are very similar, the optimized behaviour is more robust with respect to the initial scenario.

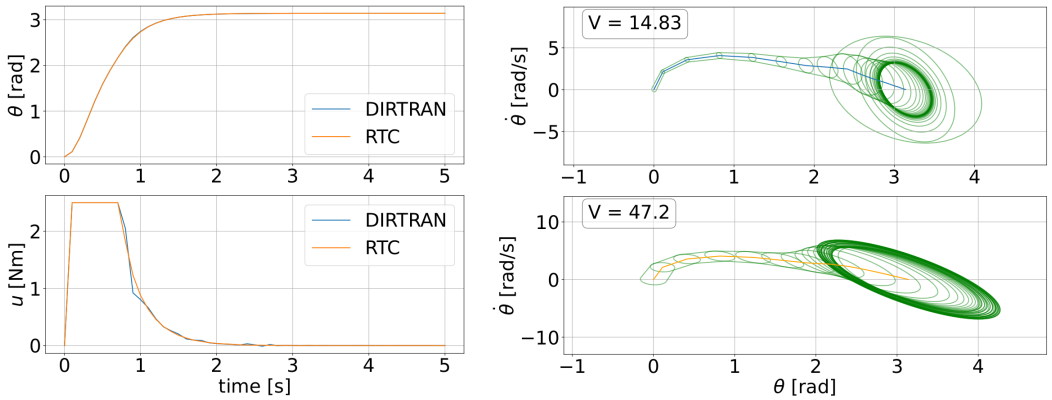

Figure 5.6: RTC vs DIRTRAN: nominal trajectories (left) and funnels (right).

The improvement of this property is even more obvious in the RTCD case, the comparison is shown in Figure 5.7. Now, the optimized trajectory turns out to be slightly different from the initial one. Less saturation in the input torque is present and the swing-up is achieved before.

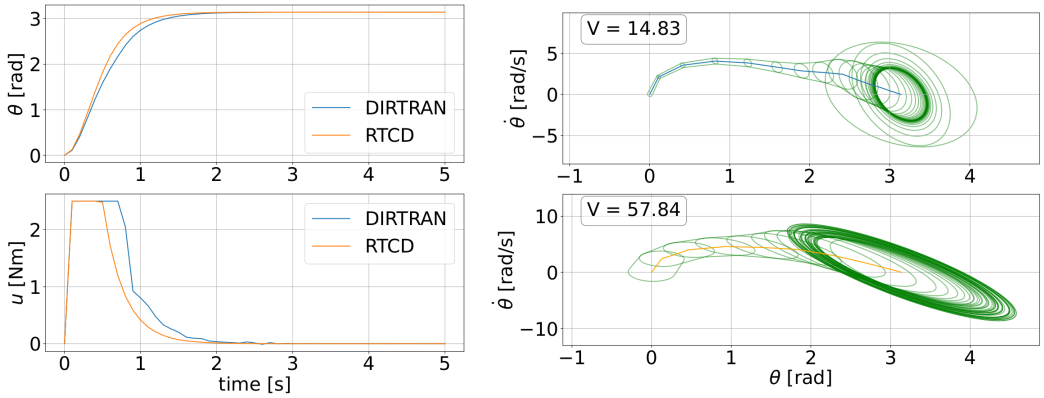

Figure 5.7: RTCD vs DIRTRAN: nominal trajectories (left) and funnels (right).

In general, the design optimization behaviour can be related to the underactuation. The best design is indeed the least underactuated. It can be noticed that in this case both the considered trajectories obtain the swing-up almost directly. However, just the otimized RTCD scenario violates the underactuation condition.
Only this last optimization case for the pendulum parameters has been considered in both simulation and experimental verification for a consistent comparison between the two frameworks.

5.1.2 Simulation Verification

The system behaviour has been modeled by using the EOM that have been described in Section 4.3.2. The state evolution has then been obtained with a simple Euler numerical integration.
An inner knot point for each optimized trajectory has been verified.The verification process firstly samples initial conditions within the ROA defined for the verified knot point. Then, each initial condition is simulated until the end of the nominal time. If the simulated trajectory stays inside the funnel for the whole of it's evolution it is considered successful. This is because other behaviours would not be compliant with the definition of funnel.

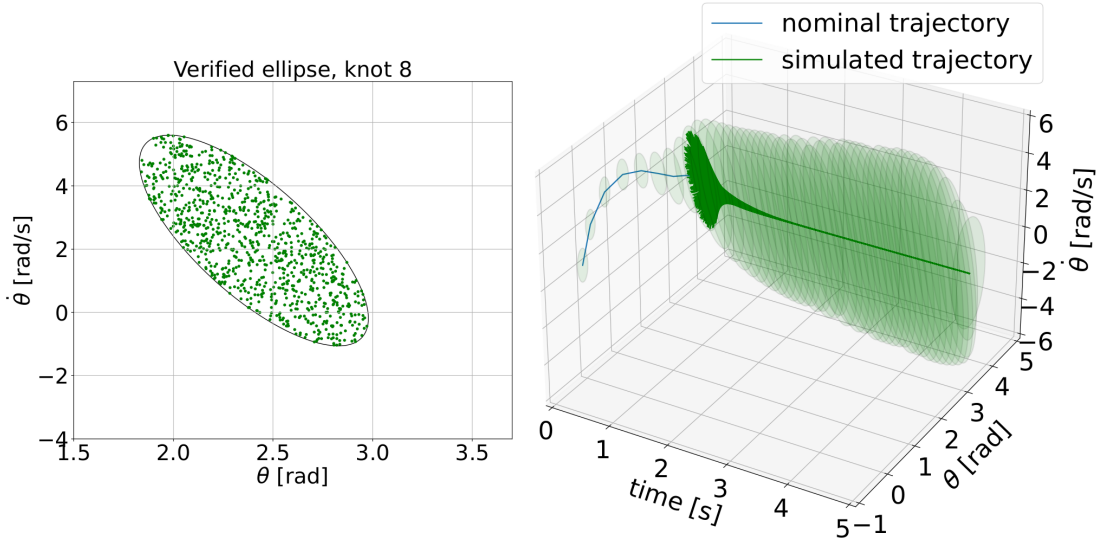

Figure 5.8: RTC inner-knot verification.

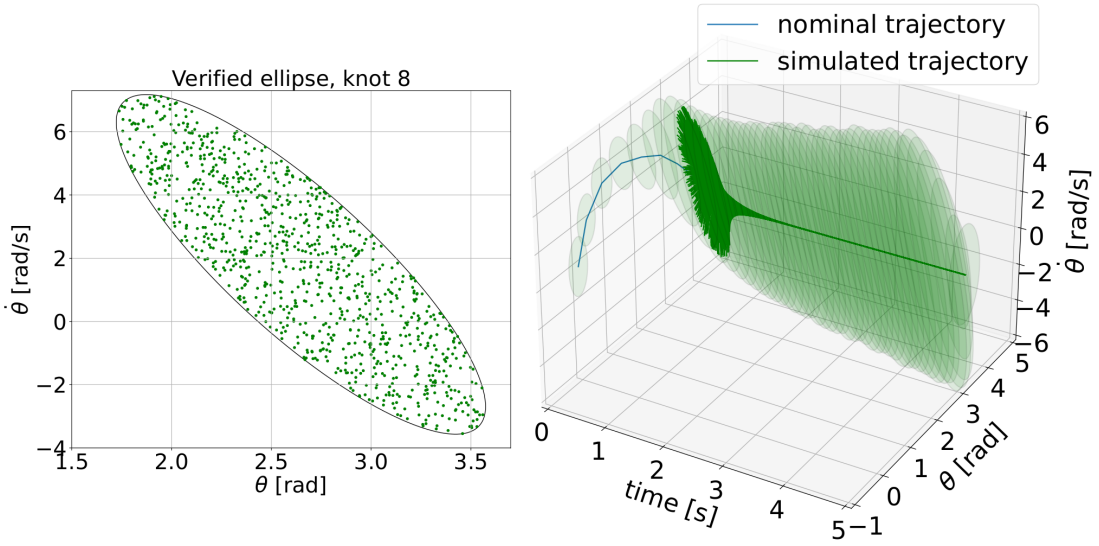

Figure 5.9: RTCD inner-knot verification.

Both optimizations result in a fully successful verification. This is expected since the SOS estimation method is providing us a formal guarantee of stabilizability, an inner estimation of the unknown real ROA. The simulated trajectories are shown in Figure 5.8 and 5.9, where they are plotted along the related funnel. All of them are indeed able to recover the off-nominal start and to stay inside the funnel until the goal achievement.

5.1.3 Experimental Verification

The choices described in Section 5.1.1 were very useful to easily implement a real verification. In particular, the selected hardware and control parameters correspond to the ones depicted in Tables 5.5 and 5.6. It can be noticed that $m_{DIRTRAN} = m_{RTC} = 0.7kg$ and that $m_{RTCD} \approx 0.6$. A weigth of $0.5kg$ have been mounted on the provided real system, hence obtaining a total mass $m \approx 0.6kg$ including link and mounting screws. This is the configuration that we have tested for the design optimized case, i.e. the RTCD case. A further weight of $m_+ = 0.1kg$ has then been attached to the center of the link in order to match the requirements of the other two cases. The hardware modification can be seen in Figure 5.10.

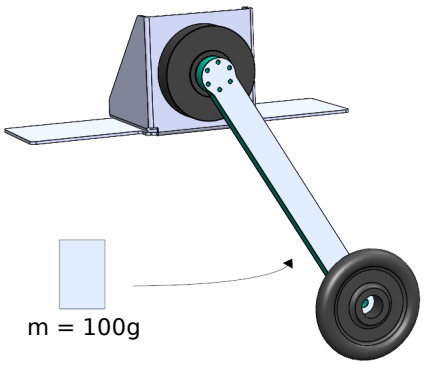

Figure 5.10: Pendulum hardware modification through the additional weight m_+.

The length of the alluminium rod remains the same for both the initial and the optimized design, i.e. $l_{DIRTRAN} = l_{RTC} = l_{RTCD}$. On the other hand, the control matrices are different in each case:

$$\mathbf{Q}_{DIRTRAN} = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{Q}_{RTC} = \begin{bmatrix} 9.5 & 0 \\ 0 & 1.2 \end{bmatrix}, \quad \mathbf{Q}_{RTCD} = \begin{bmatrix} 9.98 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_{DIRTRAN} = \begin{bmatrix} 0.1 \end{bmatrix}, \quad \mathbf{R}_{RTC} = \begin{bmatrix} 1.64 \end{bmatrix}, \quad \mathbf{R}_{RTCD} = \begin{bmatrix} 3.387 \end{bmatrix}$$

An interesting real experiment has been proposed in this section. In order to compare the robust performances betweeen the initial and final set-up, we have decided to introduce a torque disturbance. An artificial disturbance has been

addeded via software to understand the system sensibility to this kind of non-ideality. In practice, in the time window $t \in [0.5, 0.7]$ the torque input will be decreased by 3 Nm, i.e. $\tau = \tau_{des} - 3\ Nm$. The resulting behaviours are grouped in Figure 5.11.

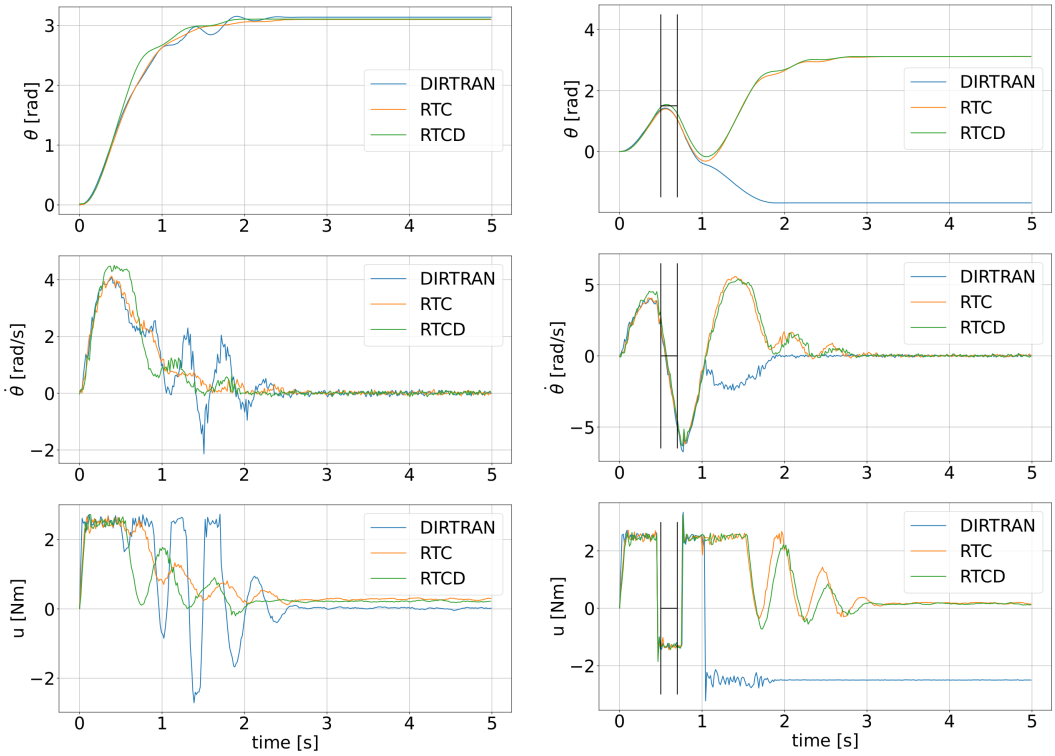

Figure 5.11: Experimental responses for different scenarios with (right) and without (left) torque disturbance.

It can be noticed in the figure above that the experiment without disturbances is successful in every case. Just a slightly less smooth behaviour is present in the $t \in [1, 2]$ for the DIRTRAN set-up. This already gives us some intuition about the improved robustness of the optimized frameworks. This idea is then confirmed in the disturbed scenario where the initial design is not able to achieve the given task.

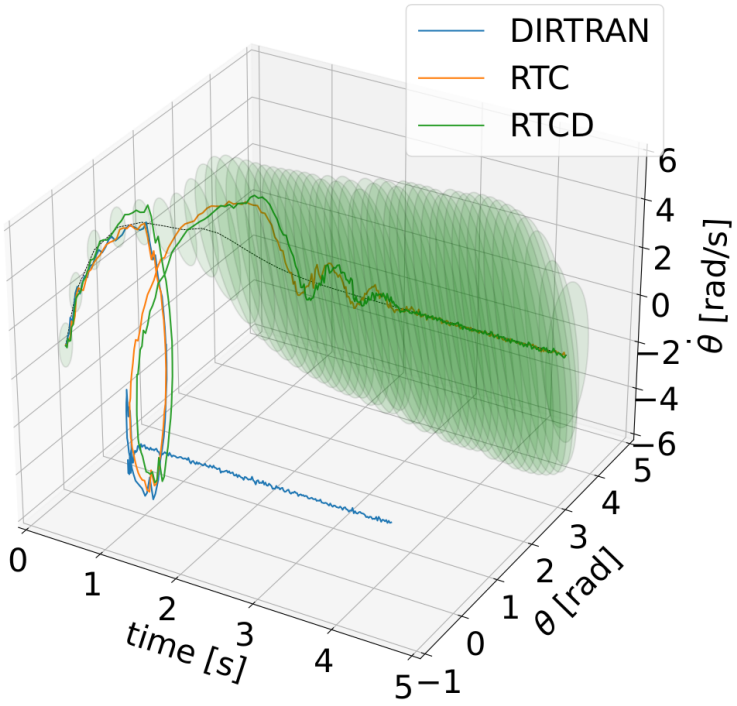

Figure 5.12: Disturbed experimental trajectories in the RTC funnel.

An interesting comparison has also been studied through this experiment, which is shown in Figure 5.12. The experimental trajectories have been plotted with respect to the RTC funnel, which is the smallest of the optimized ones. The knot-point that we have verified via simulation is exactly the one where the torque disturbance has been introduced in the experimental verification. The failing DIRTRAN trajectory exits from the stabilizable region and it is not able to recover the desired behaviour, i.e. to come back inside the funnel. It can be noticed that also the successfully stabilized trajectories are exceeding the funnel boundaries. We argue that the SOS inner ROA estimation does not give us info about the states outside the funnel. Hence, we have considered the improved recovery as a success for showing an improved general robustness.

5.2 Cart-Pole

A more complex system has been used to further analyze our implementation: the Cart-pole. The hardware design from Quanser [41] has been used for experimental verifications. Hence, also the simulations has been developed thinking about this system, which is shown in Figure 5.13. The "Linear Servo Base Unit" consists of a cart driven by a DC motor, via a rack and pinion mechanism, that ensures consistent and continuous traction. The cart is equipped with a rotary metal shaft to which a free turning pendulum is attached.

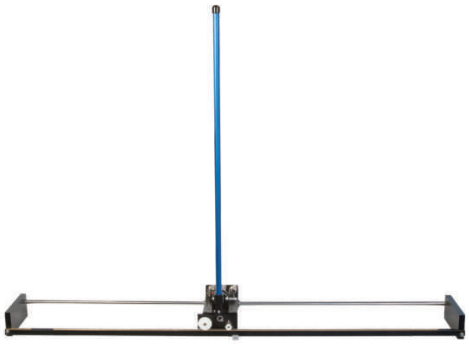

Figure 5.13: Real Cart-pole system.

5.2.1 Optimization Results

Firstly, a careful tuning process for the DIRTRAN trajectory optimization step and of the solver parameters was necessary. Then, the two proposed algorithms have been tested on this new system. The same computational tool and settings considered in Section 5.1.1 has been considered. A different estimation method for the ROA analysis has been used. The region around the goal was still obtained via SOS but the funnel around the trajectory was computed via a simulation-based method. This method has been previously introduced in Section 2.4. The most satisfying results and the related settings are listed in Table 5.7 and 5.8.

CMA-ES Init	X_0	$[\, q_{11},\, q_{22}, r\,] = [10, 10, 10]$
	σ_0	3
	$bounds$ (high)	$[20, 20, 15]$
	$bounds$ (low)	$[1, 1, 5]$
	$maxfevals$	100
	fixed design	$[\, m_p,\, l\,] = [0.227, 0.178]$
	volume	6.97
RTC Opt	parameters	$[\, q_{11},\, q_{22}, r\,] = [1.71, 1.01, 5.01]$
	timing	86 min
	volume	53.84

Table 5.7: RTC optimization settings and results: Cart-pole

CMA-ES Init	X_0	$[\, m,\, l] = [0.227, 0.178]$
	σ_0	0.1
	$bounds$ (high)	$[0.3, 0.3]$
	$bounds$ (low)	$[0.127, 0.178]$
	$maxfevals$ (outer)	1
	$maxfevals$ (inner)	40
	volume	6.97
RTCD Opt	parameters	$[\, m,\, l,\, q_{11},\, q_{22}, r\,] = [0.129, 0.185, 1.31, 1.023, 5.71]$
	timing	14 h 44 min
	volume	71.09

Table 5.8: RTCD optimization settings and results: Cart-pole

The increased state dimensionality and the different ROA estimation method makes the overall optimization to be more computationally expensive with respect to the pendulum case. The cost evolution is depicted in Figure 5.14. It can be noticed the improvement of the funnel volume, which goes up to 10 times the initial value. This improvement gives us an intuition about an increased robustness. We have noticed that most of the volume is due to the last ellipse, the one obtained via SOS. This is because the simulation-based method that we used for estimation the funnel seems to shrink very quickly as long as the considered knot point goes far from the goal state.

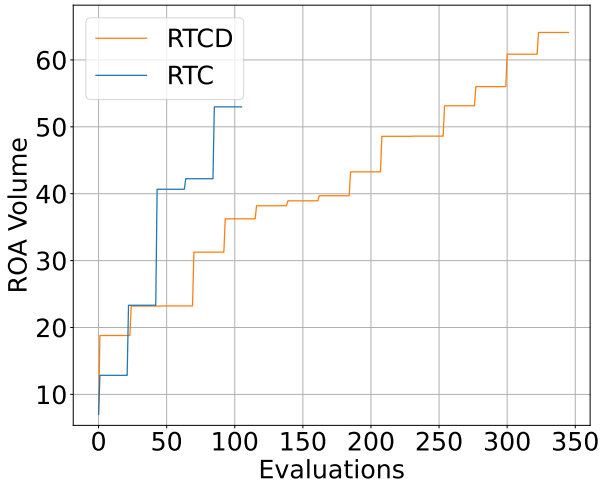

Figure 5.14: Optimization cost evolution for cart-pole.

In Figure 5.15, a comparison between initial and final ellipses around the goal state has been shown. Some meaningful slices of the state-space have been considered.

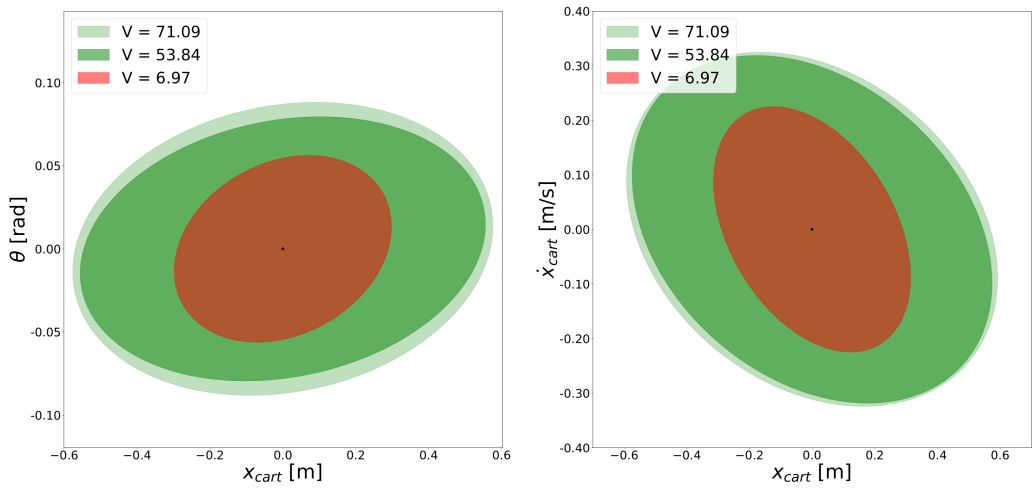

Figure 5.15: Ellipses around the goal state improvement wrt the initial shape.

5.2.2 Simulation Verification

As for the previous system, the simulated verification has been implemented by exploiting the EOM and a simple numerical integration (Forward Euler).

However, now we don't have any stability guarantee for the funnel because we are not using SOS. The verification of the improved performances has then been implemented in a probabilistic way. A set of initial states have been sampled from a grid around the rest position. The states evolution has been computed until the end of the nominal trajectory. Eventually, a simulation has been considered successful iff the simulated state ends-up in the last ellipse.
A success rate over 100 simulations has been computed for each optimized scenario to get a meaningful comparison. The initial states has been sampled in the middle of the trajectories.

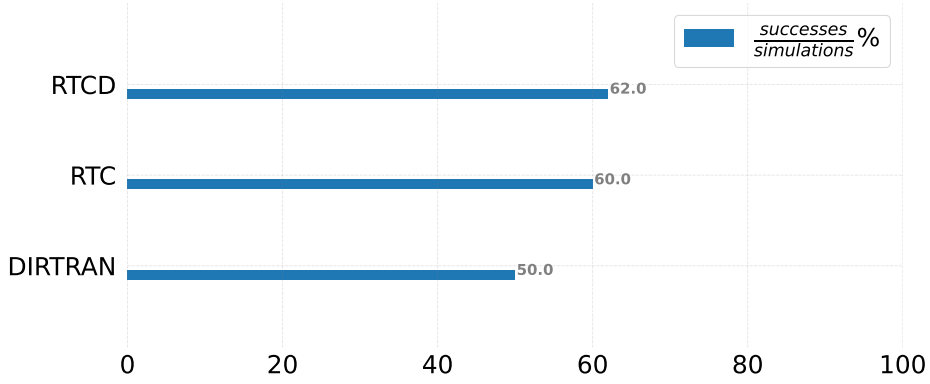

Figure 5.16: Simulation stabilization rate of off-nominal states.

As expected, in Figure 5.16 we can notice that the optimized scenarios are in general able to better recover from off-nominal initial states.

5.2.3 Experimental Verification

A real experiment has also been implemented on the Quanser linear inverted pendulum system [41] available in the laboratory of DFKI. As before, the experiment has been formulated in order to be consistent with the simulation-based verification and taking into account the possible real harware modifications. The optimized weight is almost the weigth of the pole alone, $m_p \approx 0.127$. A further weight has then been attached to the pole in order to match the weight in the non-optimized scenarios, $m_p \approx 0.227$. The used control matrices are different in each case:

$$\mathbf{Q}^r_{DIRTRAN} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad \mathbf{Q}^r_{RTC} = \begin{bmatrix} 1.71 & 0 \\ 0 & 1.01 \end{bmatrix}, \quad \mathbf{Q}^r_{RTCD} = \begin{bmatrix} 1.31 & 0 \\ 0 & 1.023 \end{bmatrix}$$

$$\mathbf{R}_{DIRTRAN} = \begin{bmatrix} 10 \end{bmatrix}, \quad \mathbf{R}_{RTC} = \begin{bmatrix} 5.01 \end{bmatrix}, \quad \mathbf{R}_{RTCD} = \begin{bmatrix} 5.71 \end{bmatrix}$$

where $\mathbf{Q}^r$ are the reduced form of the 4 by 4 matrices $\mathbf{Q}$ that include the two additional diagonal elements $q_{33} = q_{44} = 100$.

The implemented experiment consists in a torque disturbance that has been introduced in the middle of the nominal trajectory. The non-ideality has been introduced in the same knot point that we have verified before via multiple simulations. In practice, in the interval $t \in [2.51, 2.54]$ the input torque is zeroed, i.e. $\tau = 0\ Nm$. Even if both RTC and RTCD responses have been tested, only the first one was able to provides us satisfying results. The change of the hardware parameters may have been not introduced correctly or not precisely enough to implement the optimized case.

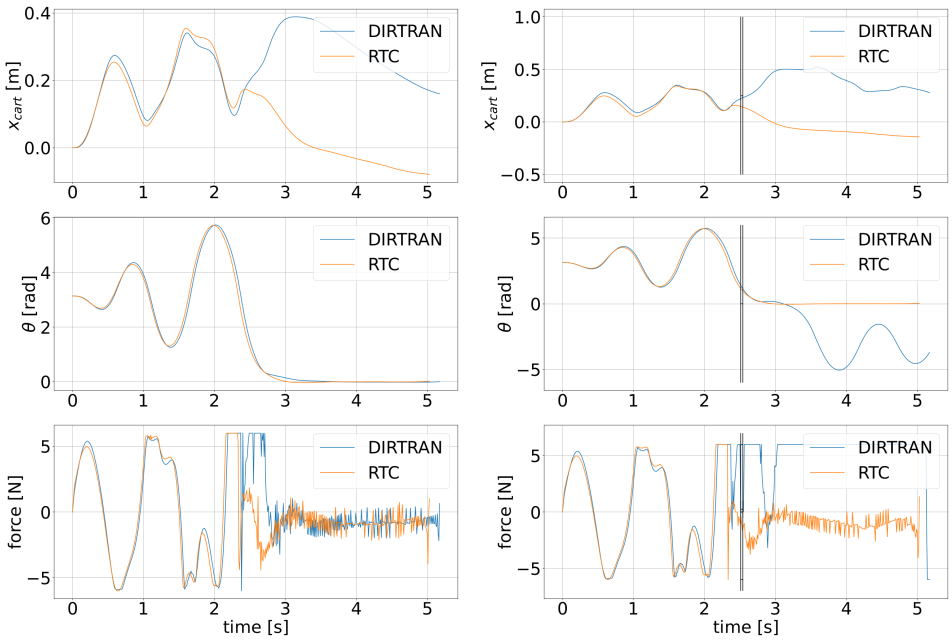

Figure 5.17: Real responses to a disturbed scenario: DIRTRAN and RTC.

As shown is Figure 5.17, the optimized set-up is now able to reject the disturbance that made the initial settings to fail. The trajectories behaviour is almost the same until the disturbance, this is the result of both closely following a similar nominal trajectory. However, the insourgence of a non-ideality caused a totally different behaviour thanks to the optimized controller.

6

Conclusions & Future Works

In this thesis work, an analysis of a sampling-based co-design process for underactuated robots is presented. Two novel algorithms have been developed to achieve a shared optimality among the design, trajectory, and stabilizing controller. Unlike the approach described in [13], both algorithms directly manipulate the stabilization cost matrices. Although this choice introduces additional complexity, it is motivated by significant improvements in the robustness cost, even when keeping the design parameters fixed. Furthermore, this choice enhances the modularity of the overall algorithm. An additional enhancement of this work is the incorporation of robustness analysis through ROA estimation. This approach offers an intuitive perspective and can potentially provide a stabilizability guarantee for off-nominal states around the optimal trajectory, particularly when employing the SOS-based estimation method. Notably, this type of estimation was not implemented in [10], where the focus was on considering the volume of a sample-based region around the desired goal. Moreover, in this thesis work, a time-varying ROA estimation is utilized as the metric for the co-optimization algorithms. Our approach has shown an improvement of the *sim-to-real gap* both in simulated and in real experiments for the two considered underactuated systems. During the experimental part of our analysis we have noticed some particular characteristics of our method. In the Pendulum case, the design optimization always pushes the system away from underactuation. Also, it resulted that having a bigger cost for the control input in the TVLQR stabilization usually improves the robustness feature. In the case of the Cart-pole system, we encountered certain challenges due to the increased system complexity. As

a consequence, we did not manage to implement the time-varying SOS ROA estimation method. Alternatively, we resorted to a simulation-based method, which still yielded improvements in robustness for the simpler co-optimization algorithm introduced. However, it is important to note that the resulting time-varying region exhibited a non-ideal shrinking behavior, indicating that the improvement primarily affected the goal region rather than the entire trajectory. Additionaly, the RTCD real verification was not achieved successfully. In our opinion, a deeper analysis on this last system and, in general, on how to best deal with the increase of the system's complexity should follow this thesis work. An improvement in the computational time is expected with a C++ implementation and an increased use of parallelization. A careful look should be kept on the SOS stabilizability guarantee, modeling and linearization errors could always affect the estimation. The proposed simulation-based approach could give advantages since it does not require the system linearization. Furthermore, this last method could be easily parallelized for improved computational performances. As a further improvement, it would be valuable to explore the possibilities of extending our co-design approach to an online setting. We believe that existing implementations of the LQR-tree algorithm [27–29] could serve as a source of inspiration for this endeavor. Additionally, extending the RTCD scheme to accommodate multiple tasks simultaneously could offer a powerful solution for more complex robotic systems.

References

[1] Boston Dynamics. *Atlas Gets a Grip*. 2023. URL: https://www.youtube.com/watch?v=-e1_QhJ1EhQ.

[2] Tesla. *Testa AI Day*. TeslaBot reveal. 2022. URL: https://www.youtube.com/watch?v=ODSJsviD_SU.

[3] Agility Robotics. *ProMat*. Digit working autonomously. 2023. URL: https://www.youtube.com/watch?v=FhlUI2xDxdE.

[4] Apptronik. *Astra Compilation*. Astra working with humans. 2023. URL: https://www.youtube.com/watch?v=FhlUI2xDxdE.

[5] Mahdi Javadi et al. "AcroMonk: A Minimalist Underactuated Brachiating Robot". In: *IEEE Robotics and Automation Letters* 8.6 (June 2023), pp. 3637–3644. DOI: 10.1109/lra.2023.3269296. URL: https://doi.org/10.1109%2Flra.2023.3269296.

[6] Felix Wiebe et al. "RealAIGym: Education and Research Platform for Studying Athletic Intelligence". In: *Robotics Science and Systems Workshop Mind the Gap: Opportunities and Challenges in the Transition Between Research and Industry*. New York, July 2022.

[7] Felix Wiebe et al. "An Open Source Dual Purpose Acrobot and Pendubot Platform for Benchmarking Control Algorithms for Underactuated Robotics". In: *IEEE Robotics and Automation Magazine (RAM)*. under review. 2023.

[8] Milan Čoh et al. "Kinematics of Usain Bolt's maximal sprint velocity". In: *Kinesiology* 50 (Jan. 2018), pp. 100–101. DOI: 10.26582/k.50.2.10.

[9] Frank Permenter Scott Kuindersma and Russ Tedrake. "An efficiently solvable quadratic program for stabilizing dynamic locomotion". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, 2014, pp. 2589–2594.

[10] Lasse Maywald et al. “Co-optimization of Acrobot Design and Controller for Increased Certifiable Stability”. In: (July 2022). DOI: 10.13140/RG.2.2.36436.07043.

[11] Zachary Manchester and Scott Kuindersma. “DIRTREL: Robust Trajectory Optimization with Ellipsoidal Disturbances and LQR Feedback”. In: *Robotics: Science and Systems (RSS)*. 2017. URL: https://github.com/HarvardAgileRoboticsLab/drake/tree/dirtrel.

[12] G. Fadini et al. “Computational design of energy-efficient legged robots: Optimizing for size and actuators”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9898–9904. DOI: 10.1109/ICRA48506.2021.9560988.

[13] Gabriele Fadini et al. “Simulation Aided Co-Design for Robust Robot Optimization”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 11306–11313. DOI: 10.1109/LRA.2022.3200142.

[14] C Semini et al. “Design of HyQ - a hydraulically and electrically actuated quadruped robot”. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 225.6 (2011), pp. 831–849. DOI: 10.1177/0959651811402275. eprint: https://doi.org/10.1177/0959651811402275. URL: https://doi.org/10.1177/0959651811402275.

[15] Q. Li, W.J. Zhang, and L. Chen. “Design for control-a concurrent engineering approach for mechatronic systems design”. In: *IEEE/ASME Transactions on Mechatronics* 6.2 (2001), pp. 161–169. DOI: 10.1109/3516.928731.

[16] James Allison and Sam Nazari. “Combined Plant and Controller Design Using Decomposition-Based Design Optimization and the Minimum Principle”. In: vol. 1. Jan. 2010. DOI: 10.1115/DETC2010-28887.

[17] James T. Allison, Tinghao Guo, and Zhi Han. “Co-Design of an Active Suspension Using Simultaneous Dynamic Optimization”. In: *Journal of Mechanical Design* 136.8 (June 2014). 081003. ISSN: 1050-0472. DOI: 10.1115/1.4027335. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/136/8/081003/6225103/md_136_08_081003.pdf. URL: https://doi.org/10.1115/1.4027335.

[18] Gabriel Bravo-Palacios, A. Del Prete, and Patrick M. Wensing. "One Robot for Many Tasks: Versatile Co-Design Through Stochastic Programming". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1680–1687. DOI: 10.1109/LRA.2020.2969948.

[19] Gianluigi Grandesso et al. *Exploring the limits of a hybrid actuation system through Co-Design - Technical Report*. Technical Report. University of Trento, June 2020. URL: https://hal.science/hal-02737086.

[20] Sehoon Ha et al. "Task-based limb optimization for legged robots". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2062–2068. DOI: 10.1109/IROS.2016.7759324.

[21] Traiko Dinev et al. *A Versatile Co-Design Approach For Dynamic Legged Robots*. 2022. arXiv: 2103.04660 [cs.RO].

[22] Nikolaus Hansen et al. *CMA-ES/pycma: r3.2.0*. Version r3.2.0. Feb. 2022. DOI: 10.5281/zenodo.6300858. URL: https://doi.org/10.5281/zenodo.6300858.

[23] Mohammmad Nabi Omidvar and Xiaodong Li. "A Comparative Study of CMA-ES on Large Scale Global Optimisation". In: vol. 6464. Dec. 2010, pp. 303–312. ISBN: 978-3-642-17431-5. DOI: 10.1007/978-3-642-17432-2_31.

[24] Durgesh Salunkhe et al. "An efficient combined local and global search strategy for optimization of parallel kinematic mechanisms with joint limits and collision constraints". In: *Mechanism and Machine Theory* 173 (July 2022), p. 104796. DOI: 10.1016/j.mechmachtheory.2022.104796.

[25] Antonios E. Gkikakis and Roy Featherstone. "Robust Analysis for Mechanism and Behavior Co-optimization of High-performance Legged Robots". In: *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. 2022, pp. 752–758. DOI: 10.1109/Humanoids53995.2022.9999745.

[26] Rizzi A. A. Burridge R. R. and Koditschek D. E. "Sequential composition of dynamically dexterous robot behaviors". In: *International Journal of Robotics Research* 8.18 (1999), pp. 534–555.

[27] Joseph Moore, Rick Cory, and Russ Tedrake. "Robust post-stall perching with a simple fixed-wing glider using LQR-Trees". In: *Bioinspiration and Biomimetics* 9.2 (May 2014), p. 025013. DOI: 10.1088/1748-3182/9/2/025013. URL: https://dx.doi.org/10.1088/1748-3182/9/2/025013.

[28] Russ Tedrake et al. "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification". In: *The International Journal of Robotics Research* 29.8 (2010), pp. 1038–1052. DOI: 10.1177/0278364910369189. eprint: https://doi.org/10.1177/0278364910369189. URL: https://doi.org/10.1177/0278364910369189.

[29] Philipp Reist, Pascal Preiswerk, and Russ Tedrake. "Feedback-motion-planning with simulation-based LQR-trees". In: *The International Journal of Robotics Research* 35.11 (2016), pp. 1393–1416. DOI: 10.1177/0278364916647192. eprint: https://doi.org/10.1177/0278364916647192. URL: https://doi.org/10.1177/0278364916647192.

[30] C. Mastalli et al. "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[31] Esmaeil Najafi, Robert Babuska, and Gabriel Lopes. "A fast sampling method for estimating the domain of attraction". In: *Nonlinear Dynamics* 86 (Oct. 2016). DOI: 10.1007/s11071-016-2926-7.

[32] Russ Tedrake. *Underactuated Robotics. Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. 2023. URL: https://underactuated.csail.mit.edu.

[33] Seth Hutchinson Mark W. Spong and M. Vidyasagar. *Robot Modeling and control*. John Wiley and Sons, 1989. Chap. 6.3, Equations of Motion, pp. 200–202.

[34] Waltar Murray Philip E Gill and Michael A Saunders. "SNOPT: An SQP Algorithm for Large-scale Constrained Optimization". In: *SIAM Review* 47(1):99-131 (2005).

[35] Lukas Gross et al. "Analytic Estimation of Region of Attraction of an LQR Controller for Torque Limited Simple Pendulum". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. 2022, pp. 2695–2701. DOI: 10.1109/CDC51059.2022.9992856.

[36] Mark M. Tobenkin, Ian R. Manchester, and Russ Tedrake. *Invariant Funnels around Trajectories using Sum-of-Squares Programming*. 2010. arXiv: 1010.3013 [math.DS].

[37] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, p. 655.

[38] Shen Shen and Russ Tedrake. "Sampling Quotient-Ring Sum-of-Squares Programs for Scalable Verification of Nonlinear Systems". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 2535–2542. DOI: 10.1109/CDC42340.2020.9304028.

[39] Nikolaus Hansen et al. "Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009". In: *ACM-GECCO Genetic and Evolutionary Computation Conference*. pp. 1689-1696. Portland, United States, July 2010. URL: https://hal.science/hal-00545727.

[40] Russ Tedrake and the Drake Development Team. *Drake: Model-based design and verification for robotics*. 2019. URL: https://drake.mit.edu.

[41] Quanser. *Linear Servo Base Unit with Inverted Pendulum*. Simulink Courseware. URL: https://www.quanser.com/products/linear-servo-base-unit-inverted-pendulum/.

[42] Felix Wiebe et al. "Torque-limited simple pendulum: A toolkit for getting familiar with control algorithms in underactuated robotics". In: *Journal of Open Source Software* 7.74 (2022), p. 3884. DOI: 10.21105/joss.03884. URL: https://doi.org/10.21105/joss.03884.

Acknowledgments

This work is the culmination of my two-year Master's program in *Control System Engineering* at the University of Padua. I am grateful for the opportunity to have been inspired by the many professional figures I encountered during this period. In particular, I would like to acknowledge the important contribution of my first examiner, Augusto Ferrante. The knowledge I gained from the various university exams has prepared me to tackle real-world applications in the field of robotics. During my time at the Bremen (DE) DFKI research institute, led by Frank Kirchner, I was introduced to this fascinating research field. The team of the *Underactuated Lab* provided me with a warm welcome and supported me for almost a year. I would like to express my gratitude to my supervisors, Shivesh Kumar and Lasse Maywald, who closely monitored my progress and provided valuable feedback when needed.

All of these accomplishments would not have been possible without the financial and emotional support of my family. Additionally, I would like to thank my friends who helped me overcome the challenges I faced and achieve a desirable work-life balance.