Department of Industrial Engineering

Doctoral School in Materials, Mechatronics and Systems Engineering
XXXV *Cycle*

Reinforcement Learning and Trajectory Optimization for the Concurrent Design of high-performance robotic systems

Supervisors:
Prof. Andrea Del Prete
Prof. Patrick M. Wensing

PhD Candidate:
Gianluigi Grandesso

2023

Abstract

As progress pushes the boundaries of both the performance of new hardware components and the computational capacity of modern computers, the requirements on the performance of robotic systems are becoming more and more demanding. The objective of this thesis is to demonstrate that concurrent design (Co-Design) is the approach to follow to design hardware and control for such high-performance robots. In particular, this work proposes a co-design framework and an algorithm to tackle two main issues: i) how to use Co-Design to benchmark different robotic systems, and ii) how to effectively warm-start the trajectory optimization (TO) problem underlying the co-design problem aiming at global optimality.

The first contribution of this thesis is a co-design framework for the energy efficiency analysis of a redundant actuation architecture combining Quasi-Direct Drive (QDD) motors and Series Elastic Actuators (SEAs). The energy consumption of the redundant actuation system is compared to that of Geared Motors (GMs) and SEAs alone. This comparison is made considering two robotic systems performing different tasks. The results show that, using the redundant actuation, one can save up to 99% of energy with respect to SEA for sinusoidal movements. This efficiency is achieved by exploiting the coupled dynamics of the two actuators, resulting in a latching-like control strategy. The analysis also shows that these large energy savings are not straightforwardly extendable to non-sinusoidal movements, but smaller savings (*e.g.*, 7%) are nonetheless possible. The results highlight that the combination of complex hardware morphologies and advanced numerical Co-Design can lead to peak hardware performance that would be unattainable by human intuition alone. Moreover, it is also shown how to leverage Stochastic Programming (SP) to extend a similar co-design framework to design robots that are robust to disturbances by combining TO, morphology and feedback control optimization.

The second contribution is a first step towards addressing the non-convexity of complex co-design optimization problems. To this aim, an algorithm for the optimal control of dynamical systems is designed that combines TO

and Reinforcement Learning (RL) in a single framework. This algorithm tackles the two main limitations of TO and RL when applied to continuous-space non-linear systems to minimize a non-convex cost function: TO can get stuck in poor local minima when the search is not initialized close to a "good" minimum, whereas the RL training process may be excessively long and strongly dependent on the exploration strategy. Thus, the proposed algorithm learns a "good" control policy via TO-guided RL policy search. Using this policy to compute an initial guess for TO, makes the trajectory optimization process less prone to converge to poor local optima. The method is validated on several reaching problems featuring non-convex obstacle avoidance with different dynamical systems. The results show the great capabilities of the algorithm in escaping local minima, while being more computationally efficient than the state-of-the-art RL algorithms Deep Deterministic Policy Gradient and Proximal Policy Optimization. The current algorithm deals only with the control side of a co-design problem, but future work will extend it to include also hardware optimization.

All things considered, this work advanced the state of the art on Co-Design, providing a framework and an algorithm to design both hardware and control for high-performance robots and aiming to the global optimality.

Acknowledgements

First and foremost, I gotta give a huge shout-out to who have been my investors from day 0, a.k.a. my family. Besides for financing me for a good amount of time, Mom, Dad, you guys deserve a standing ovation for putting up with my classic Sunday lunch complaints and, nonetheless, for always trying to encourage me and push me forward. A particular thanks goes to my sister, for not stressing me out with tons of questions about how the PhD was going. And a special thanks also to my grandma, who have been keeping me fueled with her legendary weekend coffees.

Now, let's talk about my incredible girlfriend, Alice. She's a real-life superhero, juggling a million things while still managing to be patient with me when I'm a nervous wreck. I don't know how she does it, but I'm forever grateful for her unwavering support and understanding. You're a keeper, babe!

To my wild bunch of friends from my hometown, you guys are absolute legends. I mean everyone, from the longtime JDP mates like Tommy and Albi to the legendary Morbidelli nuts as Albe, Gio and Pino. We've had way more than some insane parties, epic road trips, and ridiculous adventures together. You're the reason I know how to have a good time. Thanks for always being there, even when things got wild and crazy!

And let's not forget my equally bonkers friends from Trento. We turned that city upside down with our shenanigans, didn't we? From legendary social dinners and BBQs to epic random booze-ups, you guys made my time in Trento a non-stop party. Cheers to the unforgettable memories!

A very special shout-out goes to all my De Campi flatmates, Ivan, Sara, Luca and Ema, as well as Ale and Filippo for the short stay in France. You guys are the real MVPs. Our place has been a sanctuary of friendship, freedom and weird and way too philosophical talks. Thank you for the laughter, the support, and the countless memories we've

shared together. You've been my second family, no cap!

To my colleagues, a.k.a the lunatics I worked with at The Madhouse, you're a bunch of mad geniuses, starting from the bomb Edo. You probably have no clue how insanely challenging it's been for me to work amid all that chaos and noise, like working in a rage room, but it's been definitely worthy. Thanks for the laughter, the madness, and the endless supply of bullshit. I wouldn't trade those crazy moments for anything.

Lastly, I need to give a special shout-out to the FWR startup crew, captained by our beloved stakhanovite Marco. You guys are the dream team, full of passion and innovation. Together, we tackled challenges, rocked presentations, changed thousands plans, and we will keeping doing it until our billionaire exit. Thanks for giving me the kick in the butt to keep hustling hard and for reminding me that work can be both crazy and fulfilling.

To everyone who has been a part of my PhD journey, whether mentioned here or not, thank you for being the incredible characters in my life story. Your support, laughter, and shared adventures have made this journey one hell of a ride. I owe you all a round of drinks!

And to my academic mentors and advisors, especially Andrea and Patrick, thanks for not losing your minds dealing with me and for always listening to my complaints. Your guidance, expertise and wisdom have been invaluable and they have shaped me into the researcher I am today.

In conclusion, this thesis is dedicated to each and every one of you crazy, wonderful people. You've made this PhD journey unforgettable, and I wouldn't have survived without you. Cheers to the madness, the laughter, and the memories that will last a lifetime!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis mainly deals with *concurrent design* (*Co-Design*), which is a novel technique that takes account of both hardware and control in the design process to simultaneously optimize them and achieve top-level performance. It is structured into five chapters and its goal is to show the effectiveness of Co-Design, study its main limitations and propose some methods to overcome them. More precisely, this introductory chapter 1 begins by presenting the classic techniques used to design robots, with a particular focus on the two most popular tools for optimizing the control strategy. The introduction ends with a table classifying the state-of-the-art co-design frameworks present in the literature and with the list of contributions of this thesis. The second chapter 2 presents an application of Co-Design [1] which shows its efficacy in assessing the performance of a redundant actuation system in terms of energy efficiency. This study highlights the main problem affecting Co-Design which is treated in the next chapter. The problem concerns the ease of getting stuck in poor local optima when the co-design problem is highly non-convex and characterized by many different solutions. This is faced in chapter 3, which proposes an algorithm to learn a good control policy to be used as initial guess provider for the co-design problem and escape from poor optima aiming at global optimality [2]. This thesis ends with chapter 4 by drawing the conclusions of the study carried on so far and presenting the possible future directions that it could take.

1.1 Classic Robot Design

Robotics has a long story. The term *robot* was coined for the first time by the Czech play-writer Kerel Čapek when he wrote his *Rossum's Universal Robots* in 1920. From a cultural perspective though, we can find the roots of robotics already in the Greek mythology, with legends like that of Titan Prometheus or that of the giant Talus enacting the ambition of human beings of giving life to their artifacts. Čapek used the term *robot* to express a more specific and modern concept, which actually is the aim of robotics nowadays: Rossum built an automaton to substitute human beings in subordinate labor duties - indeed, *robota* means executive labor in Slav languages. However, that automaton did not precisely embody the image that people have nowadays of a robot because it was made with organic material. We have to wait until the 1940s for the robot to be conceived as a mechanical character, when the famous science fiction writer Isaac Asimov referred to it as a mechanical automaton with a programmable brain similar to a human being but without the capability to experience and express feelings. It was Asimov himself to use for the first time the term *robotics* to indicate the science of studying robots. He also proposed the *three fundamental laws* that form the basis of this science:

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey the orders given by human beings, except when such orders would conflict with the first law.
- A robot must protect its own existence, as long as such protection does not conflict with the first or second law.

These additional specifications regarding the behavior of a robot integrated the standard design process followed until then by engineers and specialized technicians, which considered robots simply as common industrial products. Disregarding the fact that in the collective imagination robots have an anthropomorphic aspect, considering robots as common industrial products is erroneous. In fact, the underlying peculiar feature of robots is the capability to modify the environment in which they operate: based on intrinsic rules of behavior and on data acquired on its state and on the environment, a robot performs actions and consequently change both its state and that of the environment. This

is the reason why robotics is also defined as the science that studies the intelligent connection between perception and action.

It is clear that a robot is not a simple product, but rather a complex *mechatronic system* whose design process requires skills from different fields, such as mechanics, electronics, control and computer science. Let us make some practical examples. The classical design process of a robot starts with the definition of its mechanical structure, so by defining its desired mechanical properties (*e.g.,* stiffness, thermal resistance, *etc.*) and limits (*e.g.,* concerning weight, size, *etc.*). Then it continues with the selection of the actuation system to provide the robot with the capability to move its mechanical components to perform the necessary actions to accomplish specific tasks (*e.g.,* manipulation, locomotion, *etc.*). This involves also the selection of the sensory system to enable the robot to perceive both its state (*proprioceptive sensors*) and that of the environment (*exteroceptive sensors*). Finally, to connect perception to action, a control architecture is designed (involving *modeling* and *planning*) and implemented (requiring programming skills at different levels).

1.1.1 Robots classification

Robots can be classified either based on their mechanical structure or considering their functional requirements. From a mechanical standpoint, the basic distinction that can be made is between robots with a fixed base, called *robot manipulators*, and those with a mobile base, namely *mobile robots*. Whereas, from a functional perspective, robots can be divided into two main categories: *industrial robots*, used in industry and characterized by high execution repeatability and accuracy, and *advanced robots*, featuring a high level of *autonomy* needed for operating in *unstructured* environments.

Robot manipulators

A *robot manipulator* is a sequence of rigid bodies (*links*), starting from a fixed base, connected by elements named *joints* that allow the relative motion of the links. This sequence can be either an *open kinematic chain*, when the two ends of the chains are connected by only one sequence of links, or a *closed kinematic chain* if there are multiple sequences of links connecting two joints. Each joint provides the robot with a single degree of freedom (DoF), which can be either translational (*prismatic* joints) or rotational (*revolute* joints). The number of DoF is equal to the number of joints in robot manipulators with an open kinematic structure,

whereas it is lower when the kinematics is closed because of the constraints modeling each loop. A manipulation task requires six DoF to be performed because the manipulator's extremity, consisting in a *wrist* and an *end-effector*, needs three DoF to position an object in 3D space and other three DoF to orient it. If the manipulator has more DoF than the variables that the task requires to set, then the kinematics of the manipulator is defined as *redundant*.

One of the key metrics to evaluate the manipulator's capability to reach objects is the *workspace*, that is the part of 3D space accessible by the manipulator's end effector, whose shape and volume depend on how links and joints are distributed along the manipulator's *arm*, as well as on the joint limits. The type and sequence of joints composing the arm are considered to further classify robot manipulators as:

- *Cartesian*: three prismatic joints with mutually orthogonal axes that enclose a rectangular parallel-piped workspace and enable straight motions in space.
- *cylindrical*: revolute joint at the base and two prismatic joints that cover a portion of hollow cylinder as workspace.
- *spherical*: the first two joints are revolute while the last one is prismatic and its workspace is a portion of hollow sphere.
- *SCARA*: stands for Selective Compliance Assembly Robot Arm and is characterized by one prismatic joint and two revolute joints disposed to have all the axes of motion parallel to each other.
- *anthropomorphic*: three revolute joints with the first joint axis orthogonal to the axes of the other two joints that are instead parallel to each other. Its workspace is approximately a portion of a sphere.

All types of manipulators listed above have an open kinematics. These robots are suited for manipulation tasks of relatively lightweight objects. Whereas, when a task requires to position a large payload, closed loops kinematics manipulators are the best option because of their higher mechanical stiffness which also enables high operational speeds. Closed-kinematics chains can be created by adding links to open-kinematics manipulators in such a way that parallelogram geometries can be identified in their structures. However, the mechanical stiffness gain often implies a reduction of the workspace, such as in *parallel* manipulators that have their base connected to the end-effector by means of several kinematic chains.

Mobile robots

The other class of robots is that of *mobile robots*, that are able to move freely in the environment thanks to their mobile base. Typically, these robots are used for exploration and inspection tasks rather than for tasks requiring manipulation skills. Nonetheless, in the recent years companies (*e.g.* in the oil&gas sector) are promoting the development of mobile robots with manipulation capabilities to enable full automation in their production plants. Mobile robots can be further divided into two categories based on their *locomotion* system:

- *Wheeled* or *tracked* robots: the base is connected to the ground by means of a system of (*fixed*, *steerable*, *caster*) wheels. They can also have tracks over the wheels and carry *trailers*.
- *Legged* robots: they consist in multiple rigid bodies connected to each other to form various mechanical structures, often inspired by nature to mimic the biomechanical efficiency of animals like cheetahs and some insects. Locomotion is realized by the extremities of the kinematic chain making periodic contacts with the ground. Some examples are hexapods, quadrupeds and bipeds.

There exist also robots that merge the characteristics of both robot manipulators and mobile robots, mounting a manipulator on their mobile base and so able to perform tasks involving both manipulation and inspection. In this case though, the design process is more challenging since it is harder to achieve the mechanical balance of the whole robot considering both its static and dynamic behavior, in addition to the difficulties related to the control architecture emerging from the coupling of the two actuation systems.

Industrial robots

Industrial robots refer to the wide class of robots used in *structured* environments, that is when their characteristics can be modeled *a priori*, that typically are required to provide high task execution accuracy without an high level of autonomy. Other two important features of industrial robots are versatility and flexibility, meaning that an industrial robot is designed to fit the widest set of potential applications (*e.g.* machining, assembly, welding *etc.*) as well as to be flexible enough to cover the widest range of potential tasks either by physically replacing some components (*e.g.* end-effector tool) or by reprogramming its software

(*e.g.* to perform another motion). The general definition of a robot given by the Robot Institute of America back in 1980 is actually what an industrial robot represents nowadays: *a reprogrammable multifunctional manipulator designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks.* This set of characteristics comes from the continuously increasing need for automation over the years, which entails the replacement of human beings with robots in industrial processes for both task execution and information processing. This is the reason why, from a commercial perspective, these robots are by far the most widespread robotic application. This, in turn, explains the fact that all the seminal works that have been forming the state of the art of robot control during the years were born from the need to solve industry related problems.

In the recent years, another important need has started emerging, that is the capability of co-working between human beings and robots for automatizing only specific phases of a task (*e.g.* heavy-duty, requiring high-precision or involving repetitive and stressing operations). For this reason, part of the robotics community has started focusing on the development of collaborative robots (*cobots*).

Even though industrial robots are the most established robotic application, there are still some challenges that roboticists are facing. They can be shortly summarized as follows:

- Transition from nominal to human-friendly task specification to manage foreseen and unforeseen variations occurring while the robot is working.

- Efficient integration of manipulation capabilities into mobile platforms to enhance the performance of existing solutions and design of complete solutions combining efficiently mobility and manipulation.

- Cost reduction of the components of high-performance robots, in particular their actuation systems.

- Development of open reliable systems to be accepted for widespread use.

- Design of robots for autonomous disassembly and recycling to enable sustainable manufacturing.

Advanced robots

The two main characteristics defining advanced robots are decision autonomy and capability of physical interaction with human beings. Contrary to industrial robots, the environment where advanced robots work is not structured, meaning that there is no *a priori* knowledge on the physical and geometrical properties of the environment. Moreover, in addition to robots that perform tasks in place of or in collaboration with human operators (*field robots*), advanced robots include also robotic solutions that more generally aim at enhancing the quality of life of human beings (*service robots*). This category of robots is very broad and a thorough discussion of the existing solutions is out of the scope of this thesis. Nonetheless, the main subclasses are summarized in the following list.

- Legged robots: this family of advanced robots comprises hoppers, humanoids, quadrupeds and hexapods. Legged robots are mainly used to perform rescue, inspection and maintenance tasks in place of human beings in hazardous environments (*e.g.* nuclear plants, explosive areas, post-earthquake urban areas), in addition to other applications including logistics (*e.g.*, delivery) home assistance, and agriculture.

- Unmanned Ground Vehicles (UGVs): these are wheeled or tracked mobile platforms that are remotely controlled or teleoperated and share the same goal of legged robots. Their technology is much more mature, indeed they are very reliable and high-performance, but their use is limited to rough terrains without big obstacles that cannot be circumvented.

- Unmanned Aerial Vehicles (UAVs): commonly known as drones, these advanced flying robots have reached an high level of autonomy, almost at the level of UGVs. They were initially developed for military applications, then their range of applications rapidly extended up to recreational use.

- Autonomous Underwater Vehicles (AUVs): mostly remotely operated vehicles (ROVs) and underwater drones for inspection. Their main application concerns inspection and maintenance activities in the oil&gas industry (*e.g.* pipelines, offshore platforms).

- Hybrid robots: they combine the features characterizing two or more subclasses described in the previous points. A perfect example of such advanced robots is the wheeled quadruped [3].

Mechanical design process and criteria

Even when considering very different robots, the process to design their mechanical architecture is rather standard and is based on the functional requirements representing the family of tasks that they have to perform and that translates into hardware requirements and specifications. The different stages in the standard robot design process can be summarized as follows:

- choose the kinematic chain topology describing the mechanical structure. First, the robot type (serial, parallel, Π-joint or hybrid) is selected. Then, the type of joints of the various subchains are chosen (*e.g.* revolute or prismatic).

- determine the geometry in terms of dimensions of the links (*e.g.* by using the Denavit and Hartenberg (D-H) convention [4]) to satisfy workspace requirements.

- compute the structural dimensioning of links and joints to meet the static load requirements, considering the operation conditions that are most demanding or most likely to occur.

- do the same to meet the dynamic load requirements (inertial effects due to the movement of the links and the manipulation of objects).

- compute the elastodynamic dimensioning of the whole structure to avoid those excitation frequencies that are most likely to characterize the system under the operation conditions that are most demanding or most likely to occur.

- choose the actuators and the mechanical transmissions.

Typically, a designer follows the steps listed above sequentially, but iterations may be needed if something varies in the requirements, which is the case when the operation conditions change or when supply chain issues limit the choice of some components for instance. It is clear that this process may easily become cumbersome and very long, requiring many resources, especially if iterations are needed and they restart at the third step.

Regarding the design criteria, they are almost always dictated by the specific application and in most of the cases they aim at maximizing some metrics measuring the final performance of the robot.

The most common metrics regards the reachable workspace, which often is desired to be symmetric (*e.g.* cylindrical, rectangular or spherical) for control simplicity reasons. In other cases instead, it has to assume a specific shape, for instance for safety reasons when working in structured environments with many objects, in which case the workspace is specified as a set of desired positions for the end-effector (task space).

Another performance metrics that is commonly used as design criterium is dexterity, which is defined as the ability to move and apply forces/torques in arbitrary directions with equal ease. Clearly the goal in all cases is to maximize dexterity, which can be achieved by properly design the mechanical structure besides its control strategy.

Other criteria concern the minimization of the total weight of the robot (typical in space applications or to reduce costs), the maximization of the stiffness of the mechanical structure to guarantee good controllability of the end-effector (*e.g.* in terms of positioning accuracy), and, considering manipulators, the minimization of the generalized-inertia ellipsoid (defined as $G = J^{-T}MJ^{-1}$, where M is the inertia matrix and J the Jacobian relating the joint velocities to those of the end-effector) to facilitate controllability or lower the torque requirements.

If the goal of mechanical design criteria is the maximization of some performance metrics or the minimization of some other cost-related metrics, it sounds natural to pose the mechanical design process as a constrained optimization problem, where the constraints may represents, for instance, limits on the actuators or on the shape of the workspace. As it will be clearer later, this is not enough to produce top-level performing robots. Indeed, also control must be taken into account in the optimization process, simultaneously with the hardware, and this is exactly the premise of Co-Design.

1.2 Trajectory Optimization and Reinforcement Learning

This section deals with the control side of robot design and introduces a learning technique that gained a lot of interest in the recent years, namely Reinforcement Learning (RL), highlighting its similarities to Trajectory Optimization (TO). Such a comparison is made because these methods are very much used in the literature to solve co-design problems, as they are powerful tools to find solutions of an optimization problem. Since TO is widely used by the robotics community, this section simply re-

calls its main concepts and properties, assuming the reader is already familiar with the topic. The aim of this section is to provide the tools to better analyze the literature review on Co-Design and understand in detail the algorithm presented in chapter 3. For this reason, reviewing the RL algorithms for continuous control is out of the scope of this section, which focuses on the Deep Deterministic Policy Gradient (DDPG) algorithm [5].

1.2.1 Trajectory Optimization

Optimal control (OC) is a powerful framework for formulating control problems using the language of optimization. It has a very long history, with its origins dating back to the calculus of variations in the 17th century, and it has been used to solve control problems in many different fields, such as engineering, operations research, finance, and even social sciences. The turning point for OC was the invention of the computer in the 1950s, which provided the computational means to solve large-scale optimal control problems.

An optimal control problem (OCP) takes the following general form:

$$\begin{aligned}
&\underset{x(\cdot),\,u(\cdot)}{\text{minimize}} && \int_{t_0}^{t_f} l(x(t),u(t)) + l_{t_f}(x(t_f))dt && && \text{(1.1a)}\\
&\text{subject to} && \dot{x}(t) = f(t,x(t),u(t)) && \forall t \in [t_0,t_f] && \text{(1.1b)}\\
& && x(t_0) = x_0 && && \text{(1.1c)}\\
& && g(t,x(t),u(t)) \leq 0 && \forall t \in [t_0,t_f] && \text{(1.1d)}
\end{aligned}$$

where $l(x(t),u(t))$ is the *running cost*, $l_{t_f}(x(t_f))$ is the *terminal cost*, $f(t,x(t),u(t))$ is the system dynamics (non-linear when considering robotic systems), x_0 are the initial conditions and $g(x(t),u(t),t)$ are the *path constraints*. As it will be discussed more into detail in section 1.3, an OCP becomes a co-design problem if one considers also some design parameters ρ as decision variables in addition to $x(\cdot)$ and $u(\cdot)$, as follows:

$$\begin{aligned}
&\underset{x(\cdot),\,u(\cdot),\,\rho}{\text{minimize}} && \int_{t_0}^{t_f} l(x(t),u(t),\rho) + l_{t_f}(x(t_f),\rho)\ dt && && \text{(1.2a)}\\
&\text{subject to} && \dot{x}(t) = f(t,x(t),u(t),\rho) && \forall t \in [t_0,t_f] && \text{(1.2b)}\\
& && x(t_0) = x_0 && && \text{(1.2c)}
\end{aligned}$$

Table 1.1: Families of methods for solving OCPs

Search	**Optimize ⇒ Discretize** ***Continuous Time***	**Discretize ⇒ Optimize** ***Discrete Time***
Global	Hamilton-Jacobi-Bellman	Dynamic Programming
Local	Pontryagin Maximum Principle, Calculus of Variations, Indirect Methods	Direct Methods

$$g(t, x(t), u(t), \rho) \leq 0 \qquad \forall t \in [t_0, t_f] \tag{1.2d}$$

It is worth noticing that the solutions of an OCP, namely $x(\cdot)$ and $u(\cdot)$, are trajectories, thus infinite-dimensional objects, and that also the constraints are infinitely many. This is the reason why such problems are also known as Trajectory Optimization (TO) problems. Therefore, solving an OCP - or TO problem - must involve discretization at some point, which can be done before or after the optimization process of the decision variables. This distinction, together with the type of search that is performed, namely local or global, characterize the four families of methods that have been developed during the years for solving OCPs, as reported in Table 1.1.

This section restricts itself only to recalling some concepts of the direct methods, as they can easily handle equality and inequality constraints, which is not possible with dynamic programming that also does not scale well to high-dimensional systems, and are numerically more stable and easier to initialize than indirect methods. Moreover, direct methods are the only ones that have been used for the works presented in the next chapters.

The key idea behind all direct methods is first to discrete the OCP and then to use a non-linear programming (NLP) solver to find a local optimum that minimizes the cost function. The discretization is performed by parametrizing in some way (*e.g.* with polynomials) the state and control trajectories, and then the constraints are enforced on the discretized time grid $t_0 < t_1 < ... < t_N$. There are three types of direct methods:

- Single Shooting
- Collocation
- Multiple Shooting

Single Shooting

This is the simplest direct transcription method for TO and it works by removing $x(\cdot)$ from the decision variables and discretizing only $u(\cdot)$. Most often, the continuous control input is approximated with a piecewise function, which implies the alignment of the discontinuities with the steps of integration, but also other functions can be used, as orthogonal polynomials. Once $u(\cdot)$ is discretized on $t_0 < t_1 < ... < t_N$, the state trajectory is obtained by integrating the dynamics with some integration scheme (high-order integration schemes, as Runge-Kutta 4, are typically more efficient). Since the gradient of the cost and the Jacobian of the constraints, which depend on the state, must be evaluated to solve the NLP, the sensitivities of the integration scheme must be computed. Single shooting is suited for simple problems, whereas it struggles with more complicated problems as the relationship between the decision variables and the cost and constraints is not well approximated by the linear or quadratic model used by the NLP solver.

Collocation

Collocation is a more advanced transcription method because it discretizes both $u(\cdot)$ and $x(\cdot)$ on a fine grid $t_0 < t_1 < ... < t_M$ with a polynomial for each time interval. By doing so, the constraint representing the dynamics is replaced by M constraints ensuring the consistency between polynomials at the M collocation nodes. The fact of having also the state as decision variable allows one to initialize it randomly or with some specific techniques, which is beneficial to avoiding getting stuck in poor local minima. Collocation works well with unstable systems and, even though the resulting optimization problem is large, it is sparse, so easily solvable with sparse NLP solvers.

Multiple Shooting

Multiple Shooting discretizes the control piecewise on a coarse grid $t_0 < t_1 < ... < t_N$ and integrates numerically the dynamics in each interval, as single shooting does for the whole trajectory, starting with an artificial initial value ($N - 1$ new decision variables). Once obtained the trajectory pieces, the cost integral is computed numerically and N constraints (called *defects*) are added to the NLP to ensure continuity in the nodes by closing the gaps between the starts and ends of successive trajectory pieces. Even though dividing the state trajectory into

segments increases the number of decision variables and the number of constraints, this actually makes the optimization easier for NLP solvers.

1.2.2 Basic elements in Reinforcement Learning

Before examining in depth some of the key ideas of RL, it is worth introducing some basic concepts. There are two main elements in any RL problem:

- *Agent*: learner and decision maker that performs *actions* following its *policy* and receives *rewards* and *observations* as results of the interaction with the environment.

- *Environment*: source of *rewards* and *observations* for the *agent*. It includes all the elements that are beyond the agent's direct control.

The *reward* R_t is a scalar feedback signal received at time step t and the goal of the agent is to maximize its cumulative value, namely the *return*. Reinforcement Learning is based on the *Reward hypothesis*, which asserts *"That all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (called reward)"* [6]. The *observation* O_t is a set of information describing the environment and/or agent state and the *action* A_t is the action performed by the agent that perturbs the state of the environment and/or the agent.

Reinforcement Learning is strongly linked to the concept of *state*, which is the input of both the *policy* and the *Value function*. We call *state* S_t the set of information that describes the evolution of the interaction between agent and environment. In general, the *state* can be of two types:

- *environment state* S_t^e, which is the comprehensive representation of the environment that in general is not accessible to the agent and contains all information that defines the next reward and observation.

- *agent state* S_t^a, which is the agent representation adopted by the algorithm that, together with the *environment state*, determines the next action that the agent will perform.

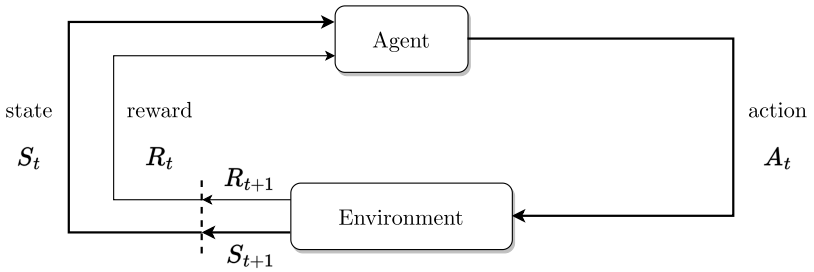


Figure 1.1: Schematic of a general Reinforcement Learning algorithm

1.2.3 Markov Decision Process

The mathematical mean used to formally define all elements constituting an RL problem is the Markov Decision Process (MDP) theory.

A *Markov Process* (MP), or *Markov chain*, is a formalization of sequential decision making that is useful for describing the evolution of the interaction between environment and agent in an RL problem. It relies on the definition of *Markov state*.

Definition 1.1. A state S_t is Markov if and only if

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \ldots, S_t]$$

This means that if a state is Markov, then it completely characterizes the environment and includes all the information about past agent-environment interaction that altogether influences the future decisions (*Markov property*).

Definition 1.2 (Markov Process)**.** A *Markov Process* is a tuple $< \mathcal{S}, \mathcal{P} >$ in which $\mathcal{S}$ is a finite set of Markov states and $\mathcal{P}$ is the state transition probability matrix.

The matrix $\mathcal{P}$ takes the following form:

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{s_1,s_1} & \cdots & \mathcal{P}_{s_1,s_n} \\ \vdots & \ddots & \vdots \\ \mathcal{P}_{s_n,s_1} & \cdots & \mathcal{P}_{s_n,s_n} \end{bmatrix}$$

where each element

$$\mathcal{P}_{s_i,s_j} = P(S_{t+1} = s_j | S_t = s_i)]$$

is the transition probability from s_i at time t to s_j at time $t+1$ with $s_i, s_j \in \mathcal{S}$.

When also a reward signal is involved in the process, the MP turns into a *Markov Reward Process*, defined as follows.

Definition 1.3 (Markov Reward Process)**.** A finite *Markov Reward Process* (MRP) is a tuple $< \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma >$ where $\mathcal{S}$ is a finite set of Markov states, $\mathcal{P}$ is the state transition probability matrix, $\mathcal{R}$ is a reward function with $\mathcal{R}_s = \mathbb{E}[R_{t+1}|S_t = s]$ and $\gamma \in [0, 1] \subset \mathbb{R}$ is a discount factor.

Definition 1.4. The *return* G_t is the total discounted reward from time step t onward

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The introduction of the *return* aims to achieve a far-sighted evaluation of each state, while the discount factor γ is useful from a mathematical point of view to avoid infinite returns in cyclic Markov processes. Therefore, one may opt for a "myopic" approach in the state evaluation using a very small γ, or a more "far-sighted" one if γ tends to 1. The expectations in Def. 1.3 and Def. 1.4 are needed when there are sources of stochasticity affecting the interactions between agent and environment. In this case, an expectation is used also to compute the *Value function*, defined as follows.

Definition 1.5. The *Value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}[G_t|S_t = s]$$

In other words, the Value function measures the expected long-term value of being in a certain state s (at time t in an episodic setting with finite terminal reward).

The Value function defined in Def. 1.5 can be decomposed in two terms:

$$\begin{aligned} v(s) &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |S_t = s] && (1.3)\\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1})|S_t = s] && (1.4)\\ &= \mathcal{R}_s + \sum_{s' \in \mathcal{S}} \gamma \mathcal{P}_{s,s'} v(s') && (1.5) \end{aligned}$$

In (1.4) the first term R_{t+1} is the immediate reward and the second one $\gamma v(S_{t+1})$ represents the discounted value associated to the next state S_{t+1}. (1.4) is called *Bellman equation* and it can be expressed in matrix notation as follows:

$$\bar{v} = \bar{r} + \gamma \mathcal{P} \bar{v} \tag{1.6}$$

where $\bar{v} = [v(s_1), \ldots, v(s_n)]^T$ and $\bar{r} = [\mathcal{R}_{s_1}, \ldots, \mathcal{R}_{s_n}]^T$
(1.6) can be solved analytically with respect to the Value function:

$$\bar{v} = (I - \gamma \mathcal{P})^{-1} \bar{r} \tag{1.7}$$

However, this implies the knowledge of the transition probability matrix, which is usually unknown, and requires a matrix inversion operation, whose computational complexity grows as $O(n^3)$. Therefore, iterative methods are the only feasible way for solving large MRPs.

A last step is needed to pass from an MRP to a Markov Decision Process (MDP) and it involves the modeling of an agent able to perform actions that affect the next state.

Definition 1.6 (Finite Markov Decision Process)**.** A finite *Markov Decision Process* (MDP) is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ where $\mathcal{S}$ is a finite set of Markov states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ is the state transition probability matrix with probabilities $\mathcal{P}^a_{s,s'} = P[S_{t+1} = s' | S_t = s, A_t = a]$, $\mathcal{R}$ is a reward function with $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$ and $\gamma \in [0, 1] \subset \mathbb{R}$ is the discount factor.

Definition 1.7. A *policy* π is a distribution over actions given states

$$\pi(a|s) = P[A_t = a | S_t = s]$$

It is worth noticing that if the policy is deterministic, then it boils down to a direct mapping $\pi : S \to \mathcal{A}$.

Given an MDP $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ and a policy $\pi(a|s)$, the sequence $(S_1, S_2, \ldots, S_n)$ is a Markov Process $< \mathcal{S}, \mathcal{P}^\pi >$ and the sequence of states and rewards $(S_1, R_1, S_2, R_2 \ldots, S_n, R_n)$ is a Markov Reward Process $< \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma >$. In case of a stochastic policy, the probability matrix $\mathcal{P}^\pi$ and reward function $\mathcal{R}^\pi$ are redefined as follows:

$$\mathcal{P}^\pi_{s,s'} = \sum_{a \in A} \pi(a|s) \mathcal{P}^a_{s,s'} \tag{1.8}$$

$$\mathcal{R}_s^\pi = \sum_{a \in A} \pi(a|s) \mathcal{R}_s^a, \tag{1.9}$$

where $\mathcal{P}_{s,s'}^a = P\left[S_{t+1} = s' | S_t = s, A_t = a\right]$ and $\mathcal{R}_s^a = \mathbb{E}\left[R_{t+1} | S_t = s, A_t = a\right]$.

In an MDP, also the meaning of the Value function slightly changes compared to Def. 1.5, and another important component called *action-value function* needs to be defined.

Definition 1.8. The *Value function* $v_\pi(s)$ of an MDP is the expected return starting from state s and following the policy π

$$v_\pi(s) = E_\pi[G_t | S_t = s]$$

Definition 1.9. The *action-value function* $q_\pi(s, a)$ is the expected return starting from state s, performing action $a \in \mathcal{A}$, and following policy π afterwards

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$$

Thus, in an MDP, each policy is characterized by a specific Value function. The action-value function instead is useful for determining the search directions of the state-action space leading to improvements of the policy.

Consequently, the Bellman equation (1.4) can be reformulated as follows:

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \tag{1.10}$$

(1.10) is called *Bellman Expectation Equation* (BEE) and differs from (1.4) because in this case the expectation is computed considering the policy π. The BEE can be rewritten using the action-value function as follows:

$$q_\pi(s, a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a, A_{t+1} \sim \pi] \tag{1.11}$$

The last step consists in defining what is meant by *solving* an MDP, that is determining the *optimality* condition for both Value function and policy.

Definition 1.10. The *optimal Value function* $v_*(s)$ is the maximum Value function considering all possible policies:

$$v_*(s) = \max_\pi v_\pi(s)$$

Definition 1.11. The *optimal action-value function* $q_*(s)$ is the maximum action-value function considering all possible policies:

$$q_*(a, s) = \max_{\pi} q_\pi(a, s)$$

Theorem 1.1. *Considering a partial ordering among policies:*

$$\pi \geq \pi' \iff v_\pi(s) \geq v_{\pi'}(s),\ \forall s$$

For any MDP:

- *There exists one (or more) optimal policy* π_* *that is better than or equal to all other policies* π*:*

$$\pi_* \geq \pi, \forall \pi$$

- *All optimal policies lead to the optimal Value function:*

$$v_{\pi_*}(s) = v_*(s)$$

- *All optimal polices lead to the optimal action-value function:*

$$q_{\pi_*}(s, a) = q_*(s, a)$$

At this point, we can link the concepts of optimal Value function and optimal action-value function, obtaining the *Bellman Optimality Equations*:

$$q_*(s, a) = \mathcal{R}_s^a + \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \tag{1.12}$$

$$v_*(s) = \max_{a \in \mathcal{A}} q_*(s, a) \tag{1.13}$$

By exploiting (1.11) and (1.13) and considering Th. 1.1, we obtain a criterion to characterize an optimal policy π_*. (1.13) expresses the *Bellman principle of optimality* which plays a crucial role in any RL algorithm.

To conclude, given these definitions, our task is to find an optimal policy that maximizes the Value function of an MDP.

1.2.4 Dynamic Programming

One of the most commonly used approaches for solving an MDP is *Dynamic programming* (DP) [7]. DP leverages the Bellman principle of optimality to break down the problem of finding the optimal control strategy into a sequence of simpler sub-problems, whose solutions are stored so that each sub-problem is only solved once. It is a very general method that encompasses different strategies having in common the convergence to an optimal solution in an iterative fashion based on the Bellman principle of optimality.

Policy Iteration

Policy iteration [8] is a DP algorithm that, given a starting policy π_0 as initial guess, iteratively converges to the optimal policy π_* by alternating the two phases of policy *evaluation* and *improvement*.

In the policy *evaluation* phase, given a policy π_k, the goal is computing its state Value function v_{π_k}, which means sweeping through the state space and computing the return obtained following π_k starting from each single state. Even though the value function could be computed in this way, in practice it is much more convenient from a computational standpoint to approach this computation iteratively, leveraging the Bellman principle of optimality. Considering an n-dimensional state-space and, if the environment's dynamics is known then this process boils down to solving a linear system of n Bellman equations:

$$v_\pi(s) = \mathcal{R}_s^\pi + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^\pi v_\pi(s') \quad \forall s \in \mathcal{S} \tag{1.14}$$

When the state-space is large, this computation can be done iteratively by using the Bellman equation as an update rule starting from an initial approximation v_0:

$$v_{k+1}(s) = \mathcal{R}_s^\pi + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^\pi v_k(s') \quad \forall s \in \mathcal{S} \tag{1.15}$$

The sequence of approximations v_k can be shown to converge to v_π as $k \to \infty$ under the same conditions guaranteeing the existence of v_π, namely that $\gamma < 1$ or that the the policy eventually leads to a terminal state starting regardless the initial state [9]. Once computed v_π, the algorithm proceeds with the *improvement* phase where the policy is updated by acting greedily with respect to v_π. This means that, in each

state s, the updated action a is the one that maximizes the action value function $q_\pi(s, a)$, which is known after the evaluation phase and is useful to understand which action performs best in the short term, precisely after one step of lookahead. Thus, the improved policy is obtained as follows:

$$\pi_{k+1}(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \ \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_\pi(s') \tag{1.16}$$

This alternate double-phase process is repeated with π_{k+1} used in place of π_k, unless $v_{\pi_{k+1}}(s) = v_{\pi_k}(s)$ for all $s \in \mathcal{S}$, which represents the termination condition of the algorithm.

Value Iteration

Value Iteration is a simpler DP algorithm compared to Policy Iteration, since it deals only with the Value function and retrieves the optimal policy only after reaching convergence. Indeed, it truncates the policy evaluation step of policy iteration to just one sweep of the state-space and improves the policy in the same step in which it updates the Value function. Thus, the algorithm starts with an initial guess v_0 and then it iteratively updates the Value function v_k as follows:

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_k(s') \right) \ \forall s \in \mathcal{S} \tag{1.17}$$

The sequence of v_k can be shown to converge to v_* for $k \to \infty$ under the same assumptions made for the existence of v_*. In practice, the iterations are stopped once the change of the Value function in a sweep is below a given threshold. When this condition is met, the optimal policy is simply recovered by choosing the actions that lead to the states with highest value.

$$\pi_*(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left(\mathcal{R}_s^a + \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \right) \tag{1.18}$$

Since Value Iteration does only a single iteration of policy evaluation and usually stops before converging to the actual optimal Value function, in practice it generally performs better than Policy Iteration and finds a good approximation of the optimal Value function in much fewer steps.

Asynchronous Dynamic Programming

In both *Policy Iteration* and *Value Iteration* algorithms, the updates of the Value function are carried out by sweeping the entire state space and this is the reason why they are called *synchronous* approaches. In case of large state sets these algorithms are not efficient and such computation may be prohibitively expensive from a computational point of view. To overcome this problem, *Asynchronous Dynamic Programming* algorithms update the values of states not systematically, without following a specific order and skipping some states at each update. This means that some states could be updated more frequently than others or using old values from previous iterations, although the correct convergence is ensured only if all states continue to be updated.

The asynchronous approach is much more flexible than the synchronous one, as, for example, it enables the customization of the learning procedure by avoiding specific states that are beyond interest to focus instead on other regions of interest of the state space. Another example is the *prioritized sweeping*, which prioritizes those states whose value estimate is far from the true value by defining an update order that ranks the states based on the *Bellman error*:

$$E_b(s) = \left| \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a - \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_k(s') \right) - v_k(s) \right| \tag{1.19}$$

Generalized Policy Iteration

At this point, it is clear that Policy Iteration, Value Iteration and Asynchronous dynamic programming can be seen as three declination of a single general approach for solving MDPs: the *Generalized Policy Iteration* (GPI) method. This term simply refers to the general idea of letting the two phases of policy evaluation and policy improvement interact, independently of the details of the two processes, as the coverage extent of the state space at each update or the number of iterations for each policy evaluation. At the beginning, the two phases may seem to pull in different directions, meaning that updating the policy to act greedily with respect to the Value function usually makes the Value function incorrect for the new policy, and updating the Value function consistently with the new policy implies that the policy is no longer greedy. In the long term though, the interaction of the two processes lead to a single common solution, namely the optimal value function and optimal policy. In practice, most of the RL algorithms can be classified as GPI

frameworks. Another key to understanding the differences among GPI algorithms is the trade-off between exploration and exploitation. If, for instance, more steps of evaluation are performed, which means more updates of the Value function, then exploration is encouraged since more knowledge about the environment is acquired before updating the policy. Or, if the frequency of policy updates is increased, exploiting the Value function and maximizing the outcome by acting greedily, then exploitation is favored. The same trade-off exists in the choice between synchronous and asynchronous updates, where deciding to select more frequently some particular states based on specific criteria is clearly a more exploitative approach. In any case, since we are dealing with a finite state space under the assumption of finite MDP, the convergence to an optimal solution is guaranteed, while the rate of convergence is strongly affected by the trade-off between exploration and exploitation.

1.2.5 Monte Carlo and Temporal-Difference

The algorithms presented so far can be used to find an optimal behavior only if the environment is known, precisely its state transition probability matrix $\mathcal{P}^a_{s,s'}$. If such information is not available, the DP principle can still be exploited to find an optimal policy splitting the problem into two phases: the approximation of the optimal Value function (*Prediction*), and the estimation of the optimal policy (*Control*).

This section introduces the two main techniques based on the acquisition of experience samples through interactions between agent ad environment used by the RL community to approximate the Value function, namely *Monte Carlo* (MC) and *Temporal Difference* (TD).

Monte-Carlo Prediction

The Monte-Carlo approach for estimating the value function is probably the most straightforward way for learning from experience. In this case the environment is considered to be episodic, which means that each episode has finite length. The estimation of the value of a state s is done by considering the mean return of different episodes starting from s: as their number increases, the average return will converge to the expected value given by Def. 1.5.

Two types of estimations can be distinguished: *First-Visit* MC, which takes into account only the first time a state is entered in an episode, and *Every-visit* MC, that considers each time the same state is visited in

the episode. In both cases, it can be shown that the estimates converge quadratically to the actual state Value function as the number of visits tends to infinity [10].

MC can be used also to estimate the *action-value function*, performing the policy rollouts not only starting from all states but also for all possible initial actions. Thus, every state-action pair must be visited, which is not guaranteed in general. One way to achieve this is simply making each new episode start from a random state-action pair so that every pair has non-zero probability to be chosen.

Since MC is a general method that does not require any particular assumption on the environment, it can be applied also when the environment is known as an alternative to DP or even if it is *non-stationary*, namely when $\mathcal{P}^a_{s,s'}$ and $\mathcal{R}^a_{s,s'}$ change in time.

It is worth noting that the MC estimate of the value of a state does not depend on the estimated value of the successor state as in the case of DP, in other words MC does not perform *bootstrapping*.

Temporal-Difference Prediction

Temporal-Difference is a fundamental approach used in several RL algorithms that combines ideas from MC and DP. As MC, its learning process is experience-driven and does not need to model the dynamics of the environment. Similar to iterative DP instead, it bootstraps, so it exploits previous estimates to create targets towards which the Value function is updated. There are many different variants of TD based on the selected number of lookahead steps. The update rule of the simplest TD method, namely *TD(0)* or *one-step TD*, is as follows:

$$v_{k+1}(s) = v_k(s) + \alpha\big[\mathcal{R}^\pi_s + \gamma v_k(s') - v_k(s)\big] \tag{1.20}$$

where $\alpha \in (0, 1]$ is the step size and the term $\mathcal{R}^\pi_s + \gamma v_k(s')$ is called *TD target*, which is also used to define the *TD error*:

$$\delta_t = \mathcal{R}^\pi_s + \gamma v_k(s') - v_k(s) \tag{1.21}$$

In case one selects n lookahead steps, the update rule becomes:

$$v_{k+1}(s) = v_k(s) + \alpha \left[\sum_{i=0}^{i=n-1} \mathcal{R}^\pi_{s^{i\prime}} + \gamma v_k(s^{n\prime}) - v_k(s) \right] \tag{1.22}$$

Comparison between MC and TD Prediction

By comparing MC and TD approaches, one can better understand the effects of *bootstrapping*. TD can learn online, before knowing the final outcome, and with episodes of undefined length, while MC works only on finite episodes and has to wait their end before updating and actually learning. These two alternatives lead to a different trade-off between *bias* and *variance* of the value estimates. MC estimation is not biased because the average of the returns from state s and following π is an unbiased estimate of $v_\pi(s)$; whereas, even though the *true* TD target $\mathcal{R}_s^\pi + \gamma v_\pi(s')$ is unbiased as well, the TD target $\mathcal{R}_s^\pi + \gamma v_k(s')$ used in the TD update rule is actually biased due to the initial guess for the bootstrap value. On the other hand, MC suffers from variance due to exploration much more than TD, especially TD(0), since it performs complete rollouts rather than only n steps in the environment.

1.2.6 Policy gradient methods for continuous control

The methods presented in the previous sections can be applied to MDPs with discrete state spaces, which implies that both the Value function and the action-value function can be modeled with *lookup tables*, where a scalar value $V(s)$ or q-value $Q(s, a)$ is associated to each state or state-action pair. This is not possible when dealing with very large or continuous MDP problems due to the well-known *curse of dimensionality*: the required memory for storing the value or q-value for each state or state-action pair would be too large and exploring the whole state or state-action space would be unfeasible. Thus, parameterized function approximators with parameter vector $\theta \in \mathbb{R}^d$ must be used to approximate the value functions. These methods can generalize the update for states or state-action pairs that are not explored, by modifying the parameters and consequently the shapes of the approximating functions $\hat{v}(s, \theta^v) \approx v_\pi(s)$ and $\hat{q}(s, a, \theta^q) \approx q_\pi(s, a)$. Such function approximators can actually be employed also in the algorithms presented so far, enabling their application also to partially observable problems, for which the Markov state assumption Def. 1.1 does not hold.

There are many function approximators commonly used by the *Supervised Learning* (SL) community that could be applied to RL problems for both prediction and control tasks. However, in practice, some techniques are more suitable than others considering the peculiar aspects of RL. In fact, an RL algorithm differs from the general SL framework mainly because:

- The training samples are correlated, if updates are performed online.
- When *bootstrapping*, the value function targets are non-stationary, since the targets also depend on the approximated function.
- The acquisition of the training samples may depend on the currently learned policy.

All the methods presented so far rely on the computation or approximation of the optimal Value function or action-value function to retrieve the optimal control policy, so they can be classified as *action-value methods*. This section introduces instead a family of methods that learn a *parameterized policy* that can be used to select actions without the need for value functions, which are called *policy gradient methods*. Policy gradient methods can be divided into two classes: *actor-only* and *actor-critic*. The difference regards the use or not of a *"critic"*, which models the learned Value function or action-value function, in addition to an *"actor"*, which is an explicit representation of the learned policy. In the general case of a stochastic policy, one can express the probability to take action a at time t being in state s with parameters θ^π as follows:

$$\pi(a|s, \theta^\pi) = P[A_t = a|S_t = s, \theta_t^\pi = \theta^\pi] \tag{1.23}$$

The way these methods learn the optimal policy parameters is based on the gradient of a scalar performance measure $J(\theta^\pi)$ with respect to θ^π. The goal is maximizing the policy performance, thus they update the policy parameters by approximating gradient ascent in the performance measure as follows:

$$\Delta\theta^\pi = \alpha\widehat{\nabla_\theta J(\theta^\pi)} \tag{1.24}$$

where $\widehat{\nabla_\theta J(\theta^\pi)}$ is a stochastic estimate whose expected value approximates $\nabla_\theta^\pi J(\theta^\pi)$ and α is the step size in the gradient ascent.

In most cases, the objective $J(\theta^\pi)$ is related to the concepts of Value function and action-value function. For example, when considering an episodic environment starting from a non-random initial state and a policy $\pi(a|s, \theta^\pi) = \pi_\theta(s)$, one could employ the *initial state value* $J_0(\theta^\pi)$:

$$J_0(\theta^\pi) \doteq v_{\pi_\theta}(s_0) \tag{1.25}$$

while in continuing environments or episodic ones starting from random initial states one could use the *average reward* $r_{av}(\pi_\theta)$:

$$r_{av}(\pi_\theta) \doteq \sum_{s\in\mathcal{S}} d_{\pi_\theta}(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \sum_{s'\in\mathcal{S}} \mathcal{P}^a_{s,s'}\mathcal{R}^a_s \tag{1.26}$$

where $d_{\pi_\theta}(s)$ is the state distribution under π_θ. In this case, one should maximize the *differential value*, defined as:

$$J_{diffV}(s|\theta^\pi) \doteq \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \sum_{s'\in\mathcal{S}} \mathcal{P}^a_{s,s'} \left[\mathcal{R}^a_s - r_{av}(\pi_\theta) + J_{diffV}(s'|\theta^\pi)\right] \tag{1.27}$$

Regardless the choice of $J(\theta^\pi)$, the analytical computation of $\nabla_\theta J(\theta^\pi)$ turns out to be problematic. In fact, when the policy parameters θ^π change, they also modify the state distribution $d_{\pi_\theta}(s)$, and, since the rewards depend on both the action choice and the distribution of states in which those choices are made, they also affect the final performance. The problem is that the way in which the policy modifies the state distribution is dependent on the environment and is unknown in most cases. Fortunately, the solution comes from the *Policy Gradient Theorem*.

Theorem 1.2 (Policy Gradient Theorem)**.** *For any differentiable policy $\pi_\theta(s)$ and for an objective function $J \in \{J_0, \frac{1}{1-\gamma}J_{avV}, J_{avR}\}$, the gradient of the objective function is*

$$\nabla_{\theta^\pi} J(\theta^\pi) = \mathbb{E}_{\pi_\theta}\left[\nabla_{\theta^\pi} \ln\left(\pi(a|s,\theta^\pi)\right) q_{\pi_\theta}(s,a)\right]$$

which is often reformulated as follows:

$$\nabla_{\theta^\pi} J(\theta^\pi) \propto \sum_{s\in\mathcal{S}} d_{\pi_\theta}(s) \sum_{a\in\mathcal{A}} q_{\pi_\theta}(s,a) \nabla_{\theta^\pi}\pi(a|s,\theta^\pi) \tag{1.28}$$

and in the case of deterministic policy [11]:

$$\nabla_{\theta^\pi} J(\theta^\pi) = \mathbb{E}_{\pi_\theta}\left[\nabla_a q_{\pi_\theta}(s,\pi_\theta(s) \nabla_{\theta^\pi}\pi(s,\theta^\pi))\right] \tag{1.29}$$

Thus, thanks to this theorem, $\nabla_{\theta^\pi} J(\theta^\pi)$ can be approximated as it does not depend anymore on the derivative of the state distribution.

It is worth noticing that the term $\nabla_\theta \ln\left(\pi_\theta(a|s)\right)$, which can be rewritten as $\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}$ and is often referred to as *score function* or *eligibility vector*, is the direction in the parameters space which leads to an increase in the probability of taking action a. The update in that direction is weighted by the value of taking that action and the policy improves as result of an overall increase in the likelihood of actions with higher values.

Deep Deterministic Policy Gradient algorithm

Among the policy gradient algorithms, the first one that could be applied to the case of continuous control is the Deep Deterministic Policy Gradient algorithm [5]. It will be the only policy gradient algorithm that will be introduced in this section because it is fundamental to understand the algorithm presented in chapter 3, which has been developed starting exactly from DDPG.

DDPG is a model-free actor-critic algorithm that exploits NNs as function approximators to learn control policies in continuous action spaces, also high-dimensional ones. It uses an approach similar to Deep Q-network (DQN) [12] for learning the critic representing the action-value function, thus in the same spirit as *Q-Learning* [13], while employing the deterministic policy gradient to train the actor, namely the policy itself. It features an *experience replay buffer* to store the experience *transitions* (s_t, a_t, r_t, s_{t+1}) acquired during training and two additional networks, one for the actor $\bar{\pi}(s, \theta^{\bar{\pi}})$ and one for the critic $\bar{q}(s, a, \theta^{\bar{q}})$, which are used as *target networks* and are updated slower than the actor and critic to enhance training stability.

The actor-critic approach in this case is used to efficiently extend the DQN algorithm to high dimensional and/or continuous action spaces. In fact, the policy learned by DQN is greedy with respect to the action value function, so the algorithm has to perform a maximization over actions in order to update the networks, which is unfeasible on a large or continuous set of actions.

The parameters θ^q of the critic are updated using the TD(0) target:

$$y_{TD} = r_t + \gamma \bar{q}\left(s_{t+1}, a_{t+1}, \theta^{\bar{q}}\right), \tag{1.30}$$

where $a_{t+1} = \bar{\pi}_\theta(s_{t+1})$ is the action given by the target policy. Therefore, as DQN, DDPG minimizes the mean squared error between the value predicted by the critic and the TD(0) target, namely $L_C = \left(y_{TD} - q\left(s_t, a_t, \theta^q\right)\right)^2$. From a practical perspective, to improve training stability, the experience transitions are sampled in *mini-batches* of length S from the buffer and the critic loss is computed by averaging the losses associated to each transition of the mini-batch, that is:

$$L_C = \frac{1}{S} \sum_{i=0}^{S} \left(y_{TD} - q\left(s_i, a_i, \theta^q\right)\right)^2 \tag{1.31}$$

A large mini-batch would mean a smooth update in the stochastic

gradient descent, but if it is too large then the update would be too small and the time to reach convergence would become excessively long.

The actor is updated (always in mini-batches) to maximize the expected return, thus the loss function is simply:

$$L_A = -\frac{1}{S}\sum_{i=0}^{S} q(s_i, a_i, \theta^q) \tag{1.32}$$

In this way the update is done using the sampled policy gradient:

$$\nabla_{\theta^\pi} J(\theta^\pi) \approx \frac{1}{S}\sum_{i=0}^{S} \nabla_{a_i} q\left(s_i, a_i, \theta^q\right)|_{a_i=\pi(s_i)} \nabla_{\theta^\pi} \pi(s_i, \theta^\pi) \tag{1.33}$$

Regarding the target networks, their parameters are updated by slowly tracking those of the learned networks:

$$\theta^{\bar{q}} \leftarrow \tau\theta^q + (1-\tau)\theta^q \tag{1.34}$$

$$\theta^{\bar{\pi}} \leftarrow \tau\theta^\pi + (1-\tau)\theta^\pi \tag{1.35}$$

where $0 < \tau \ll 1$ controls the delay of the target networks updates.

DDPG is an *off-policy* algorithm, meaning that the policy learned during training is different from the one used to explore the state-action space, which is called *behavior* policy. The off-policy approach enables decoupling the exploration from the policy learning process. In order to generate the behavior policy $\mu(s)$, DDPG adds some noise sampled from a noise process $\mathcal{N}$ to the currently learned policy $\pi_\theta(s)$:

$$\mu(s) = \pi_\theta(s) + \mathcal{N} \tag{1.36}$$

The choice of $\mathcal{N}$ should be carefully made based on the environment and the task to be accomplished, because it has a strong impact on the learning process since it determines which samples are stored in the buffer. Indeed, when using function approximators, the regions of the domain where the estimates are accurate depend on the data distribution, meaning that the approximation error is lower where the sample density is higher. For the sake of completeness, the pseudocode of DDPG is reported in Alg. 1.

Algorithm 1: Deep Deterministic Policy Gradient

begin
  Initialize replay memory D;
  Randomly initialize actor and critic network parameters $\boldsymbol{\theta_t}$,$\mathbf{w_t}$;
  Initialize targets parameters $\boldsymbol{\theta_t^-} \leftarrow \boldsymbol{\theta_t}$,$\boldsymbol{w_t^-} \leftarrow \mathbf{w_t}$;
  for *episode* **in** $1..M$ **do**
    Initialize random process $\mathcal{N}$;
    Observe initial state s_1;
    for *step* **in** $t = 1..T$ **do**
      $a_t = \pi(s_t, \boldsymbol{\theta_t}) + \mathcal{N}_t$;
      Apply a_t, observe state s_{t+1} and reward R_{t+1};
      Store transition $(s_t, a_t, R_{t+1}, s_{t+1})$ in D;
      $s_t \longleftarrow s_{t+1}$;
      Sample a minibatch of M random transitions $(s_j, a_j, R_{j+1}, s_{j+1})$ from D;
      $y_j = \begin{cases} R_{j+1} & \text{if } j+1 = T \\ R_{j+1} + \gamma \hat{q}\left(s_{j+1}, \pi\left(s_{j+1}, \boldsymbol{\theta_t}^-\right), \boldsymbol{w_t^-}\right) & \text{otherwise} \end{cases}$;
      Obtain $\boldsymbol{w_{t+1}}$ by minimizing $w.r.t\ \boldsymbol{w}$ $\frac{1}{N}\sum_{j=1}^{M}\left[\left(y_j - q\left(s_j, a_j, \boldsymbol{w}\right)\right)^2\right]$, with step-size α^w ;
      Obtain $\boldsymbol{\theta_{t+1}}$ by following the gradient $\frac{1}{N}\sum_{j=1}^{M}\nabla_a \hat{q}\left(s_j, a_j, \boldsymbol{w_{t+1}}\right)\nabla_\theta \pi\left(s_j, \boldsymbol{\theta}\right)$ with step-size α^θ;
      $\boldsymbol{\theta_{t+1}^-} \longleftarrow \tau\boldsymbol{\theta_{t+1}} + (1-\tau)\boldsymbol{\theta_t^-}$;
      $\boldsymbol{w_{t+1}^-} \longleftarrow \tau\boldsymbol{w_{t+1}} + (1-\tau)\boldsymbol{w_t^-}$;
    end
  end
end

DDPG: critical aspects

When applying DDPG, there are different aspects that are critical to avoid poor final results or, in the worst case, divergence in training. Indeed, this algorithm is characterized by the three elements of the so-called *deadly triad* [14], which groups the three major sources of instability and divergence: function approximation, bootstrapping and off-policy learning. As already mentioned, the effects of bootstrapping and function approximation can be limited by using an experience replay buffer and target networks.

The element that has not been addressed yet is the effect of the exploration strategy on the learning process: since the algorithm learns off-policy, there could be some issues when the distribution of states and actions stored in the replay buffer is not similar to the distribution of states and actions that the currently learned policy would produce. Indeed, when updating the parameters of the action-value function, *extrapolation errors* [15] may occur. Considering deterministic environments, the sources of extrapolation errors can be of two types: if any state-action pair (s, a) is not present in the buffer (*absent data*), then the estimated action-value for those pairs can be largely different from the real one, especially if using non-linear function approximators (*e.g.* neural networks); if the distribution of transitions in the buffer is different from the distribution of states and actions that the current policy would experience (*training mismatch*). In other words, if the exploration noise causes important changes in the state-action visitation distribution, the off-policy algorithm may achieve very poor learning performance. In an actor-critic setting, as in DDPG, where the actor is updated using the critic and viceversa, extrapolation errors may easily lead to divergence. In order to mitigate the training mismatch, the choice of an exploration noise with zero expected value and low variance may help, with the hope that in the long run the bias to the policy's actions distribution should be limited. To compensate for issues due to insufficient data, especially at the start of the training, one could consider to add an initial *warm up* phase to the training, where the buffer is filled up with experience transitions without training the networks. The aim is reducing the probability of having an initial biased estimation of the action-value function that could quickly lead to divergence of the training. This strategy, coupled with a random initialization of the state, should also decrease the danger of running into *absent data* by having transitions in the buffer related to regions of the state-action space as diverse as possible. Another

strategy that could be adopted is decoupling exploration and learning using *asynchronous updates*, that means updating the neural networks at separately from the network updates, so that to let the agent explore more before modifying the behavior policy.

In addition to exploration, another crucial aspect is the buffer sampling strategy. Since the importance of the information carried by each transition is not the same, sampling more frequently the transitions that are more informative helps speed up the learning process. A possible way to weight each transition with respect to its information is considering the associated TD error:

$$y_{TD} - q\left(s_t, a_t, \theta^q\right) \tag{1.37}$$

This strategy called *Prioritized Experience Replay* (PER) [16] aims at increasing the data sampling efficiency by replaying more often those transitions that are associated with a worse approximation of the value function, which lead to a higher learning potential. The priority p_j of transition j is defined as:

$$p_j = |y_{TD_j} - q\left(s_t, a_t, \theta^q\right)| + \epsilon \tag{1.38}$$

where ϵ is a small positive constant to ensure that each transition has non-zero probability to be sampled. The probability P_j of sampling transition j is defined as:

$$P_j = \frac{p_j^\alpha}{\sum_k p_j} \tag{1.39}$$

where α is an hyperparameter determining the level of prioritization: $\alpha = 0$ means uniform sampling, whereas $\alpha = 1$ means fully prioritization, which implies that the probability is directly proportional to the TD error. However, using a non-uniform sampling would introduce bias in the estimations, so *importance sampling* is needed to anneal that bias. It consists in weighing each TD-error by a weight w_j defined as follows:

$$w_j = \left(\frac{1}{N} \cdot \frac{1}{P_j}\right)^\beta \tag{1.40}$$

The paper introducing DDPG [5] reports that a small bias can be accepted at the start of the training, for this reason the hyperparameter $\beta \in [0, 1]$ is used to increase the compensation for the bias as the training proceeds by linearly increasing it to 1.

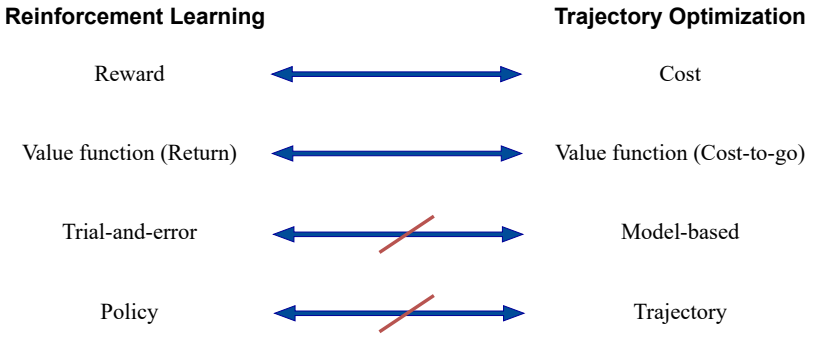


Figure 1.2: Similarities and differences between RL and TO.

Other RL algorithms for continuous control

After DDPG, other algorithms have been proposed for learning continuous control policies, most of them being based on it or taking inspiration from it. The aim of this paragraph is not to describe them, but only to provide the reader with an overview of what other options could be used to treat the same problems solvable with DDPG. For this purpose, the following list summarizes the RL algorithms for continuous control that are commonly used as benchmark:

- Soft Actor Critic (SAC) [17]
- Proximal Policy Optimization (PPO) [18]
- Trust Region Policy Optimization (TRPO) [19]
- Twin Delayed DDPG (TD3) [20]
- Asynchronous Advantage Actor Critic (A3C) and its synchronous variant A2C [21]

For the implementation details of each of these algorithms, the reader is invited to visit the GitHub repository of Stable Baselines3 [22].

1.2.7 Dualism between RL and TO

Reinforcement Learning and Trajectory Optimization may seem two different approaches, sharing some common points (e.g., the goal of solving

an OCP), but showing clear differences (e.g., the former is a trial-and error approach whereas the latter is model-based). In fact, they are two derivations of the same principle, that is the *principle of optimality* underlying DP expressed by the Bellman equation (1.4). The *principle of optimality* states that, considering an optimal N-step control sequence $\{u_0^*, ..., u_{N-1}^*\}$ leading to the state sequence $\{x_1^*, ..., x_N^*\}$ starting from state x_0, the subproblem starting from x_k^* at time k of length $N-k$ has the truncated control sequence $\{u_k^*, ..., u_{N-1}^*\}$ as optimal solution.

As shown in Fig. 1.2, RL and TO are dual to some extent. Indeed, the reward function in RL plays the same role as the cost function does in TO, with the only difference that they have opposite sign. Thus, as already mentioned, RL aims at maximizing the return, while the goal of TO is finding the minimal cost-to-go. Again, considering the same problem to be solved, return and cost-to-go are actually the same thing with opposite sign, as they both represent the Value function in the RL and TO sense respectively.

Of course the differences do not boil down to a mere change in sign of the function encoding the performance to be optimized, but there are also key differences. The first one is represented by the approach that RL and TO follow to find a solution: the former learns by trial-and-error based on data acquired through agent-environment interactions, whereas the latter exploits the *a priori* knowledge of a model of the agent interacting with the environment, therefore both the system dynamics and a structured environment must be available. The second key difference is that RL learns a policy, so a continuous mapping between state and action spaces in the case of deep RL, whereas TO finds trajectories considering specific initial states. This clearly implies that the times to find a solution for RL and TO are on two totally different scales. But concerning their applicability RL is actually faster than TO, as the learned RL policy is used offline (DNN inference) whereas the use of TO is online, for example in a *Model Predictive Control* (MPC) fashion, which means that a new trajectory must be computed for each new initial state.

Such degree of complementarity between RL and TO is one of the key concepts underlying the algorithm that will be introduced in chapter 3. Indeed, as it will be shown, the benefits of RL and TO can be combined within a unified framework to get faster to a solution of the problem of interest with also higher probability that such solution is close to be globally optimal. More specifically, the algorithm presented in chapter 3 builds upon the structure of DDPG, which has been modified accordingly to include TO as a smart and efficient guide for exploring the state-action

space in RL. In turn, RL learns how to properly set the starting point of the TO optimization process to increase the likelihood of finding lower-cost solutions, in other words, it learns a control policy whose rollouts are used to warm-start the TO decision variables of TO.

1.3 State-of-the-art Co-Design frameworks

As highlighted in section 1.1, hardware and control of mechatronic systems have always been designed separately through two distinct processes. However, these two aspects are intrinsically connected and their interplay determines the final performance of the robot. Therefore, high-performance robots are not achievable if hardware and control are addressed in two sequential phases. This is indeed the premise of Co-Design: in order to achieve the best performance in executing a certain task, a robot must be designed considering both its mechanical structure and its control law in the same design process. Moreover, this process must exploit some optimization technique to ensure that the task is accomplished in the best possible way.

This section reviews the main co-design frameworks from the literature, clustering them according to the optimization technique, and it classifies them in a table, based on a specific taxonomy.

1.3.1 Evolutionary Algorithms based

The first work regarding Co-Design dates back to 1994 in the context of computer graphics, when Sims [23] published his seminal work introducing a technique for generating virtual creatures in a simulated 3-D world based on evolution and co-evolution. He leveraged evolutionary algorithms (EA) and parallel computing to search over a huge number of potential morphologies and behaviors, with the aim of finding the right combinations defining virtual characters able to accomplish tasks such as walking, jumping, swimming, and following a light source. To do so, he used the same genetic language for both morphology and control: directed graphs of nodes and connections defining how signals flow between nodes. More precisely, each node in the graph describes a rigid part and it may contain sensors (joint angle sensors, contact sensors, photosensors) to measure some properties related to that part or to the world interacting with that part, or neurons containing a set of basic functions (*e.g.,* sum, min, if, integrate, *etc.*) that transform input signals (coming from sensors or describing an internal state) and send them

to effectors (each of them controlling a degree of freedom of a joint) or other neurons through connections to give creatures the capability to perform arbitrary actions. Given a specific task represented by a *fitness* function and starting from an initial population of genotypes, offspring are generated by copying and combining the directed graph genotypes of the survivors from the previous generation, namely those individuals whose fitness values fall within the survival percentile determined by the *survival-ratio*. Despite the incredible results, the system proposed by Sims is limited to the context of computer graphics because virtual creatures do not need to satisfy physical constraints, as long as the resulting animations look natural. Indeed, the only constraints that are considered are joint limits, but they are just a small part of the set of constraints characterizing a real robotic application. Moreover, another limitation is that rigid bodies are modeled only as parallelepipeds and this could prevent the search from finding the actual best-performing robot design. Nonetheless, this work inspired a number of subsequent publications also in robotics [24–47].

Twenty years after Sims and still in the context of computer science, in 2013, Cheney *et al.* [27] extended [23] with the aim of creating virtual characters that are more complex and with properties more similar to those of natural organisms than Sims' creatures. To do so, they evolved morphologies composed of voxels of different materials (hard or soft to mimic bones or soft tissues and inert or expandable to represent supportive tissues or muscles) and behavioral options with a compositional pattern-producing network (CPPN), that is an encoding able to generate symmetry and repetition regularities and evolved according to the NEAT algorithm [48], rather than direct encoding like in [23]. The results in terms of morphologies and behaviors found are fascinating, showcasing the power of evolutionary algorithms, but this method essentially shares the same main limitation of Sims' work, namely the limited applicability to the virtual world.

One year later, in 2014, Digumarti *et al.* [28] presented the first co-design work applied to a real robot, the quadruped StarlETH [49], that used a direct policy search method based on the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [50] to find the optimal combination of design and controller enabling StarlETH to run at high speeds. Specifically, they kept the actuators, robot symmetry, leg configuration and gait type fixed, and they optimized link lengths, torso dimensions and batteries relative position (6 design parameters) as well as parameterized trajectories of trunk and feet and some parameters of

the motion control (up to 27 gait and control parameters). Even though the results obtained in simulation are not validated on real hardware, they still demonstrate that the concurrent design of hardware and control for legged robots can substantially enhance their locomotion skills.

1.3.2 Optimal Control based

After the initial work of Sims that was EA-based and animation-oriented, researchers started developing co-design frameworks leveraging the knowledge of the model of the robot dynamics to use OC for searching the optimal hardware-control pair. The choice of using OC for Co-Design, which means considering some design parameters as decision variables in addition to state and control variables, turns out to be the most popular in the literature. This is most probably due the capability of OC to encapsulate the interaction between robot and real world in a single formulation through the definition of constraints that the final solution must respect.

The first OC-based co-design work was published in 2006, when Li *et al.* [24] proposed a concurrent engineering methodology for robotic systems, named Design For Control (DFC), based on a general mathematical formulation that models the design problem as a single constrained optimization problem. Among the four variants of DFC, only one can be properly classified as a co-design method, since the mechanical structure and the controller are jointly optimized by solving a unique optimization problem, whereas the others involve decomposing the design problem into subsequent subproblems to be solved iteratively. The main weakness of the co-design variant of DFC lies in the potential difficulty of having a mathematical model to describe such a complex design problem and, for this reason, the other three iterative variants are actually more practical for robotic systems design.

In the context of bipedal robots, in 2009, Mombaur [26] used numerical optimization to investigate the combinations of design parameters and actuation inputs leading to self-stable motions of human-like robots. Specifically, the problem is formulated as a multiphase OCP where the optimization variables are: the state variables, the control variables, the vector of phase switching times, and the vector of model parameters (length, relative CoM location, mass and moment of inertia of each segment composing the humanoid model, and the parameters defining the spring-damper element in each joint). The proposed formulation takes account also of different constraints related to the hybrid dynamics of

the robot, the design variables limits, complex relations between variables as well as switching and periodicity conditions. For solving such a complex problem, the author firstly discretizes it using a multiple shooting algorithm (MUSCOD) [51], and then solves the resulting structured non-linear program with an efficient sequential quadratic programming (SQP) algorithm.

In the same year, Allison [29] applied the OC-based co-design approach to the partial redesign of mechatronic systems in case redesigning control is not enough to meet the requirements of a new application and partial hardware (plant) design changes are needed, ensuring minimal cost modifications. More precisely, the new methodology, called Plant-Limited Co-Design (PLCD), involves an initial sensitivity analysis based on the study of the model parameter Jacobian to identify the (continuous) candidate plant modifications and reduce the PLCD problem complexity, and a final optimization (using a gradient-based method called SNOPT [52]) to minimize the redesign cost while taking account of all the system constraints. The effectiveness of PLCD is demonstrated through the redesign of a two-link planar manipulator, that was initially designed for a pick-and-place task and then is required to perform another task that is not feasible with only control changes. The results show that minimal redesign cost changes can be achieved with PLCD, whereas fully redesigning the system would imply substantial plant modifications potentially not affordable.

Going back to the context of humanoid robots, Buondonno *et al.* [31] introduced a framework to jointly optimize the state and control trajectories and some design parameters (*e.g.*, step length, stiffness of the transmission elements and drive inertia moments) of underactuated biped walkers considering three types of actuators (rigid, series elastic, and parallel elastic). The framework is based on the formulation of a single OCP considering a time horizon of a step and solved with a newer variant of MUSCOD (MUSCOD-II). Besides the capability of finding the optimal design and control given a specific actuator, the results highlight also that Co-Design is the appropriate tool to effectively analyze and fairly compare different actuator types.

In 2017 and more extensively in 2018, Ha *et al.* [32, 38] presented an iterative co-design framework extending the capabilities of their previous work [30] by taking account also of the robot morphology in the optimization process. Given a parameterized robot design and a set of end-effector trajectories, the algorithm proceeds by iteratively adjusting the optimization variables (link lengths, actuator placements, joint tra-

jectories, actuator inputs, and contact forces), taking advantage of the implicit function theorem to derive the relationships among parameters without violating the constraints. The simulation tests were performed on two systems, a 4-DoF manipulator and a small quadruped, and the results obtained in simulation for the quadruped were validated by fabricating the optimized design and applying the optimized control to finally verify the expected 30% reduction of the maximum current required.

Also the interactive framework proposed by Desai *et al.* [34] enables the optimization of both morphology and behavior of legged robots. Starting from the initial user-defined robot morphology and task (*e.g.*, desired direction and speed of motion), the motion is optimized by solving an optimization problem with quadratic cost terms encoding the tracking of the desired direction, speed of motion and pose (in terms of CoM and end-effector positions), and the constraints satisfaction. Then, the design is automatically optimized by leveraging the *adjoint method* to quickly compute the search direction and find the optimal hardware (number of links, links length, number of actuators, and actuator position) and motion parameters (represented as a function of the robot's hardware).

As shown in [31], co-design techniques can be exploited also to analyze (*e.g.*, assessing the energy efficiency) different actuators and to enable a fair comparison between them by considering the optimal combination of hardware and control for each actuator. This is precisely what Yesilevskiy *et al.* have done in [35], where they formulated and solved (using MUSCOD) a single OCP to find motion trajectories (and the time to perform a hop), actuator inputs, and hardware parameters (gear ratio and spring stiffness) that are optimal for each actuation system (parallel elastic and series elastic actuators) used in a 1D hopper.

In 2020, Bravo-Palacios [53] *et al.* proposed a co-design framework combining stochastic programming with TO to tackle the problem of scalability of multi-task co-design problems. The two-stage stochastic programming formulation (nevertheless represented by a single optimization problem) enables the decoupling of the scenarios, each of which obtained by randomly perturbing some features of the environment or task at runtime to represent uncertainty, achieving faster computation times and higher scalability than a sequential formulation. The framework was applied to a two-link manipulator required to perform three maneuvers while carrying a range of loads and a planar monoped robot asked to perform jumps in various terrains, and the results validates the capability of the optimized designs to balance performance across many

different scenarios.

More recently, in 2022, Bravo-Palacios [54] *et al.* extended their previous framework by considering also the optimization of the feedback gains to add robustness reasoning for motion planning. This is the first framework that simultaneously optimizes the nominal trajectory, the hardware parameters and the feedback controller. The results with n-link manipulators and a planar monopod robot show the benefits of treating also the feedback gains as decision variables of the optimization process in terms of robustness to external disturbances, demonstrating also that this contributes to finding nominal trajectories that require less effort for control stabilization. Despite the good results though, there remains the scalability issue due to the co-optimization of the feedback gains that limits the number of scenarios that can be assessed through optimization.

In the same year, Dinev *et al.* [44] proposed a bi-level formulation of the legged robots co-design problem, where the lower level computes the state and control trajectories (using Crocoddyl [55]) given a task and design parameters vector, and the upper level optimizes the latter (using the interior-point/direct algorithm KNITRO [56]). Contrary to [43], this approach is fully gradient-based as the gradient of the co-design metric with respect to the design parameters is needed for the gradient-based optimization in the upper level. It as been tested on a real-world co-design problem, that is the design of the 12-DoF version of the quadruped SOLO considering trotting and jumping tasks, with the design parameters vector consisting of the link lengths, attachment points of the legs, trunk width, height and depth, payload distribution, motor mass and gear ratio. This approach is shown to scale well up to 17-dimensional design vectors and it finds the same solution faster than CMA-ES-based frameworks.

Lastly, it is worth mentioning also the work of Sartore [57], which deals with the maximization of the ergonomics in the interaction between humans and humanoids when they collaborate. Specifically, the authors formulated (in CasADi [58]) an OCP to find the optimal hardware parameters (link lengths and density, CoM height) and static joint configuration of the humanoid iCub [59] considering a collaborative payload lifting task. The results show that the optimized humanoid can reach a larger set of heights while concurrently reducing its energy consumption and maintaining that of the human. Besides the typical local minima problem of OC-based frameworks, the main limitation of this work relies on the fact that only the static case is considered, meaning

that no optimal control trajectories are computed, thus leading to a design that actually does not optimize the ergonomics during the whole task execution.

It is worth stressing that, in order to avoid misleading results with OC-based frameworks, the model of the robot must be accurate enough and the equations modeling the relationships between physical quantities and/or hardware parameters must be smooth to avoid numerical instability when using gradient-based solvers.

1.3.3 Reinforcement Learning based

More recently, given the fast growing number of RL applications and the increasing computational capabilities of modern computers (*e.g.* GPUs), some researchers proposed co-design frameworks embedding RL algorithms to exploit their inherent exploratory nature to find optimal design parameters and control policies.

In this sense, Schaff *et al.* [36] paved the way in 2018 with the first co-design framework that uses deep RL in place of classic optimization techniques to jointly learn how to design and control legged robots. Their approach keeps a design distribution (Gaussian mixture model) and leverages deep RL, more precisely a proximal policy optimization (PPO) [18] algorithm, to find the optimal control policy that maximizes the expected return over the distribution of designs. As training proceeds, the distribution is shifted towards designs that perform better and the whole process ends when converging to the locally optimal pair of design and control policy. The results obtained considering three common problems of OpenAI's Gym [60] are very promising, but there is still a gap to be filled to enable the use of this method in a real use case. In addition, the main limitations of this method are the fact that the robot morphology is kept fixed and that the training time may be excessive with complex high-dimensional problems.

Similar in spirit to [36] but with a different approach, Chen *et al.* [42] presented in 2020 another deep RL co-design framework that involves modeling the hardware to be optimized as a computational graph, as done for the control policy. In this way, auto-differentiation can be used to compute gradients that can flow through the hardware policy, enabling the joint optimization of hardware and control with RL algorithms, in this case TRPO [18]. This approach is demonstrated to outperform both CMA-ES and frameworks involving the use of CMA-ES for optimizing hardware parameters and RL for learning the control policy. It has also

been validated by co-designing and successively fabricating an underactuated 3-D printed hand. Despite the promising preliminary results, it is still unknown if this framework would be able to scale when considering higher dimensional systems and more complex tasks.

In the same year but in the context of soft robotics, Schaff *et al.* [46] presented a co-optimization framework that shows how Co-Design can be leveraged to exploit compliance in robotics and achieve passive behaviors that would not otherwise be easily programmable. They proposed a modular form of model order reduction that enables the use of finite element analysis (FEA) within a multi-task RL framework (employing the soft actor-critic algorithm [17], SAC). This step is crucial for obtaining high accuracy results and, in turn, enabling zero-shot sim-to-real transfer of the resulting optimized soft robots. This approach has been validated by co-optimizing and then fabricating a legged crawling soft robot, which demonstrates its effectiveness by crawling two times faster than a similar legged soft robot designed by an expert.

1.3.4 Using a mix of optimization techniques

Not all the frameworks in the literature use a single optimization method to solve co-design problems, indeed there are many works that mix different techniques in various ways separating the optimization of hardware and control into two interconnected loops.

The first framework that is part of this category is that of Hass *et al.* [25] proposed in 2006, where numerical optimization is used to explore efficiently a six-dimensional parameter space to assess how and to what extent the mass distribution of a passive dynamic walker influences the walking speed and stability. The parameter space is formed by three hardware variables (the CoM position and the joint radius of rotation) and three control variables (the initial step length and the angular velocities between the links and the uninclined ground). The resulting optimization problem is then solved with a population-oriented simulated annealing technique (NPOSA) [61] that combines the population concept of Evolutionary Computation (EC) with Simulated Annealing (SA) to drive the search towards the global optimum.

Ten years later, in 2016, Ha *et al.* [30] presented a framework that optimizes the control and design parameters of 2D two-links and three-links legs (of monopeds and quadrupeds) in two stages: first, the motion of a simplified model (including CoM, momentum trajectory, contact positions, and contact forces) is optimized (by SQP) to minimize contact

forces; then, the leg link lengths are optimized (by CMA-ES) as well as the related full-body motions to achieve the task while minimizing the control effort.

In 2019, Pirron *et al.* [40] introduced MPERL, a tool to automatically synthesize the design and control of manipulators starting from an abstract description of the robot's kinematics containing information about actuators, joints, and sensors. To model the robot, MPERL builds a labeled graph where the vertices are local frames representing the components (predefined or user-defined), the edges connect the components, and the labeling function maps the vertices to the component's type and properties. Once modeled the robot, MPERL estimates the workspace and perform a singularity analysis, then it generates the constraints in such a way that an inverse kinematic solver (dReal SMT solver [62] in addition to least squares optimization and a cyclic coordinate descent algorithm [63]) can handle them. Finally, MPERL generates the software to move the robot based on some user-defined actuator values to be set or target positions of the end-effector to be reached. Even though the method was tested on some traditional industrial manipulator structures, its applicability remains limited to the design of small custom 3D-printable robots. It is also hard to classify it as a co-design tool, since hardware and control are designed in two distinct phases, not simultaneously, and they are not optimized.

One year later, Chadwick *et al.* [41] developed an open-source Matlab toolbox, called *Vitruvio*, for optimizing leg designs for walking robots given as inputs an initial simplified robot model, a user-defined metric (*e.g.*, energy consumption minimization) and a motion plan (generated by the single rigid body dynamics trajectory generation framework TOWR [64]). The optimization is performed using a genetic algorithm (*Global Optimization Toolbox* [65]) and parallel computing (*Parallel Computing Toolbox* [66]) to speed up the process. The framework was applied for redesigning the legs (optimizing the thigh and shank lengths) of an existing quadruped, *ANYmal* [67], as well as for designing new quadrupedal robots (optimizing link lengths, transmission ratios, and spring parameters). Vitruvio only optimizes the design of a legged robot, therefore it cannot be considered a co-design tool. If considering the whole framework though, that is the sequential use of TOWR and Vitruvio in an iterative fashion, then it can be classified as a full-fledged co-design framework.

Another interesting work, published in 2021, is that of Fadini *et al.* [43], which introduces an algorithm for the concurrent optimization of

hardware parameters and control trajectories of legged robotic systems. It has a double loop structure, where the inner loop solves an OCP (with an approach [55] based on differential dynamic programming, DDP) to find the optimal control trajectories for each individual of the population keeping fixed the hardware parameters, and in the outer loop a genetic algorithm (CMA-ES) generates the offspring from the best performing individual. The algorithm proceeds by alternating these two phases until convergence or up to meet an ending condition. It has been tested for the design of a two-joint monoped robot (leg of the existing quadruped *SOLO* [68]) required to perform a jumping task, considering the motor mass, gear ratio, link scaling factor, task completion time and control inputs as decision variables. The results show the effectiveness of this approach, however it is challenging to assess which level of computational complexity can be handled and the maximum size of the OCP solvable in a reasonable time.

One year later, Fadini *et al.* [45] added robustness properties to their previous framework [43] by introducing a simulation step with perturbations in between the two optimization levels. The results obtained with this modification clearly show that including robustness reasoning can have a crucial impact on the solution found, as highlighted by the fact that in both tests (with a 4-DoF manipulator and the same monoped robot used in [43]) the mechanical transparency of the optimal hardware suggests a trade-off between energetic optimality and the capability to react to perturbations.

Lastly, it is worth mentioning also the recent work of Belmonte-Baeza *et al.* [47], which introduces a model-free co-design framework which is used to optimally redesign the legs (thigh and shank lengths) of ANYmal. The first phase of the framework deals with control optimization and uses meta RL (employing PPO), that is an RL technique to allow quick adaptation of a policy to different tasks, to train a policy able to track random velocity commands over different irregular terrains. As the policy is learned, the latter is used in the second phase (involving the use of CMA-ES) to assess various designs and find the one maximizing a design objective. This approach has also been benchmarked against [41] and it proved to be able to find lower-cost designs.

1.3.5 Using other optimization techniques

Most of the Co-Design literature involve the use of the aforementioned optimization techniques: EA, OC, RL, or a mix of them. Nonetheless,

there are some frameworks that employ other peculiar methods to simultaneously optimize hardware and control of robotic systems.

For instance, despite its limited applicability to real use cases, it is worth mentioning the work by Schulz *et al.* [33]. They developed *Interactive Robogami*, an interactive software for the design of ground robots, based on the composition of predefined parameterized components, that can be fabricated as flat sheets to be folded then into 3D structures. It is based on a database containing both geometric parts and motions that are combined together following composition rules dictated by a specific "grammar". In its standard use, optimization is used only in the geometry manipulation (translation, rotation, and dimension scaling) to ensure feasibility of the final desired geometry. But there is also the possibility to automatically optimize (by COBYLA algorithm [69]) a user-define metrics by searching the optimal geometry within the full geometry space.

Similarly to [33] and [34], also [37] tackles the problem of simplifying and automatizing the robot creation with Co-Design. The application in this case fits into the industrial context and is very specific, that is the autonomous design of modular manipulators whose structure and program are determined based on human demonstrations. Their method consists of three steps: generation of the trajectories, selection of the assembly, and generation of the controller. To generate the trajectories, first the task is modeled using a Riemannian [70] task-parameterized (TP) GMM, which adapts context-specific demonstrations to new contexts enabling both position and orientation data handling, and then the TP-GMM is transformed into task-space coordinates and, using the product of Gaussians, a task-space distribution of trajectories (whose mean is the desired trajectory) is finally obtained. Successively, modular components (one base, three joint modules, five link modules and one end effector) are assembled by hierarchically eliminating poor-performance assemblies following the method presented in [71]. Finally, centralized model-based controllers, that are automatically generated from dynamic and kinematic parameters of the modular components, are used to automatically synthesize a passivity-based tracking controller. No optimization technique is used, but the results on real hardware show the effectiveness of this method and its applicability to real use cases.

Still in the subgroup of catalog-based co-design frameworks, Carlone *et al.* [39] in 2019 proposed a binary optimization formulation of the co-design problem, which involves the characterization of the design space and the classification of the design specifications in terms of per-

formance and constraints at a system level, and implicit constraints. A robotic system is conceived as a set of modules (*e.g.*, actuators, sensors, computational board, control algorithms, planning algorithms), and each module has a catalog of potential choices and some features (*e.g.*, cost, torque, weight, maximum speed, power consumption) describing their technical specifications. The search for the modules maximizing the system-level performance while satisfying the system-level and module-level constraints is done by solving a binary (linearized) optimization problem (by CPLEX [72]), where each binary variable indicates whether a given module has to be chosen as part of the design or not. The framework has been tested considering the design of two robotic applications, a racing drone and a team of robots for collective transport, and it was able to find an optimal design in both cases. As for all the other catalog-based co-design frameworks though, also this approach is limited by the usage of a user-defined catalog. Even if it is true that considering only commercially available components may save time and resources to designers, this potentially prevents them from finding design that could provide much better performance.

Table 1.2 classifies the aforementioned works based on the criteria listed here below:

- Formulation: whether the implementation is monolithic (hardware and control optimized in a single phase) or not. If not, how many phases it consists of.
- Application: the systems that have been used to test the framework, specifying the number of design and control variables. Additionally, a measure of the size or complexity of the problem considered (*e.g.*, number of total variables, max number of DoF, time horizon length, etc.).
- Optimization tool: if optimization is involved in the framework, the specific optimization technique/method/tool that has been used.
- Real HW: if the framework has been validated on real hardware or not. If yes, some information about the hardware.
- Limitations: main limits or potential issues of the framework.

Table 1.2: Review of the literature on Co-Design

Work	Formulation		Application				Opt. tool	Real HW	Limitation
	Monolithic	*N. of phases*	*System*	*Design variables*	*Control variables*	*Max problem size*			
Sims 1994	Yes	1	Virtual creatures	Morphology as chain of rigid bodies (parallelepipeds), parallelepiped dimensions, joint type	23 basic functions per neuron	100 generations with 300 population size	Custom genetic algorithm	No	Computer graphics context only
Li 2001	Yes (1^{st} variant), no (2^{nd}/3rd variants)	1 (1^{st} variant), 2 iterated (2^{nd} and 3rd variants)	Four-bar programmable linkage system	Link lengths, masses and CoM	PD controller gains	18 parameters	MATLAB opt. toolbox	No	Limits on optimization problem size/complexity (1^{st} variant), convergence time may be too long (2^{nd} and 3rd variants)
Hass 2006	Yes	1	2-D passive biped (inverted double pendulum)	Link CoM positions and the joint radii of gyration	initial step length and two initial angular velocities	6-D search space	NPOSA	No	Curse of dimensionality when dealing with high-dimensional spaces
Mombaur 2009	Yes	1	2-D biped (9 segments)	Segment lengths, relative CoM locations, masses, moments of inertia, joint stiffnesses, damping ratios, and offset angles (59 parameters)	7 torque control variables, 2 phase switching times	18.4k unknown variables, 17.6k equality constraints, and 36.8k inequality constraints	MUSCOD + SQP	No	A model must be available, if too complex then NLP solvers might be not able to find a solution
Cheney 2013	Yes	1	3-D soft-voxel systems	Morphology as aggregation of voxels and voxel material (4)	2 functions (expansion and contraction at pre-defined frequency) per active voxel (2 out of 4 types)	Genome size of 5000 values, 1000 generations with 30 population size	CPPN-NEAT	No	Applicability limited to soft-actuators or small soft-walkers
Digumarti 2014	Yes	1	Quadruped (ANYmal)	Link lengths (3), torso dimensions (2) and batteries relative position	Up to 27 gait and control parameters	3500 rollouts	CMA-ES	No	Convergence time may be excessively long when dealing with high-dimensional spaces
Allison 2014	No	2 (1^{st} sensitivity analysis, 2^{nd} optimization)	Two-link planar manipulator	Link lengths	Intermediate position and velocity values defining the desired trajectory	6-D search space	SNOPT	No	Morphology must be fixed, limits on optimization problem size/complexity
Ha 2016	No	2 (1^{st} motion opt., 2^{nd} design + swing foot params opt. and computation of joint related quantities, and fullbody contact forces)	Quadruped	Link lengths (12)	CoM position and orientation, contact position and forces (82)	Up to 500 iterations for the SQP solver, 100 iterations (16 parents, 32 offspring) for CMA-ES	SQP + CMA-ES	No	Convergence time may be excessively long when dealing with high-dimensional spaces, objective function for motion generation must be quadratic

Work	Formulation		Application				Opt. tool	Real HW	Limitation
	Monolithic	*N. of phases*	*System*	*Design variables*	*Control variables*	*Max problem size*			
Buondonno 2017	Yes	1	3-D biped (12 DoF)	Stiffness and drive inertia moment of series elastic actuators (10)	Step length, ground inclination angle, torque control trajectories (52)	10 multiple shooting nodes with 12-DoF state and 5 actuators	MUSCOD II	No	Limits on optimization problem size/complexity
Ha 2017	Yes	1	Quadruped	Link lengths and linear actuator attachment points (32)	Joint positions, actuator forces, and contact forces at the end-effectors (564)	596 parameters	SciPy [73]	A small-size quadruped with optimized link lengths	Desired end-effector trajectories must be defined, limited to continuous parameters only
Schulz 2017	No	Interactive GUI	Origami-inspired robots	Catalog consisting of 12 bodies, 23 limbs, and 10 peripherals each of which has a set of shape parameters and a corresponding feasible set	Gait parameters defined by the combination of parametric controllers, one per joint (prismatic, revolute, spherical) and with two phases (step, reset)	Apparently allowed unlimited combinations of components (45) and joint types (3)	COBYLA	6 robots designed for walking	Applicable to robots made up of print-and-fold parameterized components selected from a limited catalog, only one actuation choice, simulator without environmental or task constraints
Desai 2018	Yes	1	Legged robots	Link lengths, body width and length, actuator parameters (3-D attachment points for linear actuators, axis orientation for rotary ones)	Robot pose, CoM position, end-effector positions, ground reaction forces acting on them and flags indicating if they are grounded	1050 parameters (quadruped)	Custom C++ solver implementation leveraging the Adjoint method	No	Applicable to robots made of serially connected and actuated links, fixed number of links/actuators, optimization processes invisible to the user
Yesilevskiy 2018	Yes	1	Monoped	Gear ratio, spring stiffness	Joint position and velocity, motor torque, single hop time	No information about how the OCP is discretized (*e.g.*, number of knots)	MUSCOD	No	Limits on optimization problem size/complexity, fixed morphology
Schaff 2018	Yes	1	OpenAI's Gym environments	Link lengths and radii, body dimensions (up to 25)	Policy mapping continuously states to actions	GMM with 8 mixture components, up to 1.5B environment timesteps for PPO	PPO (policy) + GMM gradient-based optimization (design)	No	Convergence time may be excessively long when dealing with high-dimensional spaces, fixed morphology
Giusti 2018	No	4 (1^{st} human demonstration, 2^{nd} desired task-space trajectory generation, 3rd design generation, 4th controller realization)	Serial manipulators	Base (1), link (5), joint (3), end-effector (1) modules	Task modeling with TP-GMM, trajectory generation in task-space coordinates, passivity-based tracking controller synthesis (modified recursive Newton-Euler [74])	Search over 9.8M module combinations, pick-and-place task considering unseen target locations	TP-GMM gradient-based optimization (motion) + algorithm in [75] (module selection)	6 DoF manipulator	Limited catalog, open-chain structures only, human demonstrations needed

Work	Formulation		Application				Opt. tool	Real HW	Limitation
	Monolithic	*N. of phases*	*System*	*Design variables*	*Control variables*	*Max problem size*			
Carlone 2019	Yes	1	Robotic systems (drone, (*) team of robots for collective transport)	*e.g.*, for * frames' weight and size, sensors' power consumption, weight and size, motors' force exerted, size, weight and power consumption, and batteries' weight, size and power generated	Planning and control algorithms included in the module set	2k module combinations for *	CPLEX	No	Limited catalog, binary optimization not feasible when dealing with large catalogs, objective and constraints must be linear
Pirron 2019	No	(1^{st}) formulate the problem with MPERL language, (2^{nd}) run the software	Serial and parallel manipulators	Apparently only link lengths	An *Actuate* method (with a mapping from actuator parameters to values as input) and a *Move* method (taking a target position of the end-effector as input)	The most complex system considered is a Gough Stewart Platform	dReal SMT + least squares optimization + cyclic coordinate descent algorithm in [63]	Single and dual Scara arms	Limited catalog, applicable to the design of toy-size robots, considered only kinematic models, a generic robot description must be provided
Chadwick 2020	No	2 (1^{st} trajectory optimization with TOWR, 2^{nd} design optimization with Vitruvio)	Walking robot legs	Link lengths, transmission ratio, springs' stiffness and setpoint (additionally also leg configuration and number of links)	CoM and end-effector trajectories optimized with TOWR (given desired terrain and gait parameters)	100 generations with 150 population size considering 11 design parameters	TOWR + MATLAB Global Optimization Toolbox + MATLAB Parallel Computing Toolbox	No	Fixed morphology, an initial simplified CAD model must be available, convergence time may be excessively long when dealing with high-dimensional design spaces
Chen 2020	Yes	1	Robotic systems (mass-spring toy problem, (*) underactuated robotic hand)	*e.g.*, for * joint spring stiffnesses (4), preload angles (4) and pulley radii (4)	*e.g.*, for * the relative position setpoint for motor travel and palm position commands	*e.g.*, for * $2 \cdot 10^7$ environment steps to train a fully-connected neural network with 2 layers and 128 hidden units on each layer	TRPO	Underactuated robotic hand	Not tested with high-dimensional systems and complex tasks
Bravo-Palacios 2020	Yes	1	Two-link manipulator, monopod robot	Gear ratios (2), thigh-shin ratios for the mass and the length (2)	State and control trajectories	$\sim$28k optimization variables (monopod robot test)	PySP [76] (IpOpt [77])	No	Fixed morphology; not tested with high-dimensional systems, case in which the number of considered scenarios may be very limited
Fadini 2021	No	1 but 2 loops (outer one for design optimization and inner one for control optimization)	Robotic systems (*e.g.*, SOLO's leg testbed)	Structure scaling factor (2), motor mass (1) and gear ratio (1)	Contact phase duration, state and control trajectories (2 joints, 1000 knots)	Up to 10 evolutions of a population of 100 individuals, 10^3 trajectory optimization problems	Crocoddyl (FDDP) + CMA-ES	No	Fixed morphology, convergence time may be excessively long when dealing with high-dimensional design spaces, limits on OCP size/complexity

Work	Formulation		Application				Opt. tool	Real HW	Limitation
	Monolithic	*N. of phases*	*System*	*Design variables*	*Control variables*	*Max problem size*			
Bravo-Palacios 2022	Yes	1	Up to 7-link manipulator, monopod robot	Gear ratios (2), thigh-shin ratios for the mass and the length (2)	State and control trajectories and feedback gains	∼65k optimization variables (monopod test)	PySP (IpOpt)	No	Fixed morphology; limited scalability due to the optimization of also the feedback gains
Dinev 2022	No	2 (upper level design optimization lower level motion plan optimization)	Legged robots (SOLO quadruped)	Limbs lengths (4), leg x- and z-attachment points (4), trunk's width, height and depth, x- and z-positions of two electronics boxes in the robot's base (4), gear ratio and motor mass	State and control trajectories (12 joints, 60 knots)	17 design variables and 60 knots for 12 state and control trajectories	Crocoddyl (Box-FDDP) + KNITRO	No	Limits on OCP size/complexity
Fadini 2022	No	1 but 2 loops (outer one for design optimization and inner one for control optimization)	Robotic systems (4 DoF manipulator, (*) hopping leg)	Motor masses (for * 2), gear ratios (for * 2), link scale factors (for * 2)	State and control trajectories (for * 2 joints, 1000 knots)	Up to 10 evolutions of a population of 10^4 individuals, 10^5 trajectory optimization problems, 100 simulations per problem	Crocoddyl (DDP) + parallelized CMA-ES	No	Fixed morphology, convergence time may be excessively long when dealing with high-dimensional design spaces, limits on OCP size/complexity
Schaff 2022	Yes	1	Soft legged robots	Actuator positions (8 binary variables), pressure regulator connections (3 binary variables)	Policy mapping continuously states to actions	2048 designs, 96 parallel environments, 1M environment timesteps	SAC	Five crawling soft robots	Finite discrete design space with enumerable number of designs, training time may be excessively long when dealing with high-dimensional design spaces and/or complex tasks/environments
Belmonte-Baeza 2022	No	2 (1^{st} policy meta-training, 2^{nd} design optimization)	Legged robots (ANYmal's leg)	Link scale factors (2), gear ratios (2) (potentially also the linkage transmission geometry, leg attachment points, and 1^{st} actuator orientation)	Policy mapping continuously states to actions	2000 epochs with 1000 training environments, 30 generations with a population of 35 individuals, 300 parallel simulated environments for each individual	PPO + CMA-ES	No	Training time may be excessively long when dealing with high-dimensional design, and/or complex tasks/environments, and/or too many tasks
Sartore 2022	Yes	1	Humanoid robot (iCub)	Link lengths and density, CoM height	Static joint configuration (static joint torques computed successively)	Length multipliers and densities for torso, arms and legs of iCub	CasADi (IpOpt)	No	Fixed morphology, the presence of several local minima may prevent the solver from finding good solutions, limited to static case (no optimized control trajectories/policy)

1.4 Contributions of the thesis

The first part of the thesis presents a co-design application that shows the efficacy of Co-Design in performing an energy efficiency analysis of a redundant actuation system. Regardless the level of experience and the possible application of rules of thumb, having redundancy in the system would make it impossible for a designer to assess which combination of hardware components and control law would achieve the best performance. Moreover, the complexity in making this choice increases with the number of degrees of freedom, the number of actuated joints, the variety of joint types, the task complexity and the presence of limits on the state and/or control. The analysis introduced in chapter 2 shows that Co-Design is the suitable technique to manage such design complexity aiming also at optimality of the solution.

As the complexity of the design problem increases in terms of non-convexity of the objective function, Co-Design, or more generally TO, starts suffering from the problem of getting stuck in poor local minima. For this reason, chapter 3 focuses on this issue and introduces an algorithm that combines RL and TO for the continuous control of robots aiming at global optimality. The preliminary results are very promising and lay the foundations for a future extension to Co-Design by including also the optimization of hardware design variables besides the control. Such algorithm provides also ample room for improvement and development, such as a significant reduction of training time through the implementation of Sobolev Learning and the extension to co-design problems involving hybrid dynamics like that of legged robots.

To summarize, here is the list the main contributions of this thesis.

1. An energy-efficiency analysis of a redundant actuation system through Co-Design.

2. An algorithm that combines Reinforcement Learning and Trajectory Optimization for the continuous control of robots aiming at global optimality.

Chapter 2

Co-Design application

In the last decades, many roboticists focused their research on determining which actuation mechanisms are most suitable for robotic systems such as legged robots or industrial manipulators [78–82]. Often, the "stiffer is better" rule of thumb has been adopted as a premise of the design process. High bandwidth force control and accurate position control are the two main benefits, however to the detriment of safety in human-machine interactions and high cost of the mechanical system. While active control is able to regulate output impedance, there are fundamental limits to mechanical robustness in the case of impulsive loads. Thus, many have taken inspiration from nature, intentionally including compliance in actuation systems between the load and mechanical energy source. Some researchers [83, 84] have tried to combine different actuators to achieve both high bandwidth and high output torque, among other benefits. Their results are promising, but the characteristics in terms of energy efficiency of such actuator have not been investigated and remain unclear.

Thus, this chapter presents an energy efficiency analysis of a redundant actuation system that was possible to carry out only through Co-Design. First, standard actuators and the redundant actuation systems are introduced. Then, the methodology followed to conduct the study is described, showing how to formulate a co-design problem, and the results are reported, drawing conclusions about whether the redundant actuation system is more energetically convenient compared to standard actuators. Lastly, a variant of the co-design framework with robustness reasoning is presented, which was the result of a collaborative work led

by G. Bravo-Palacios [54] that involved the author of this thesis.

2.1 Standard actuation systems

To date, there is no actuation mechanism that uniformly outperforms the others. This is due to the strong dependency on the task (*e.g.*, walking, holding objects, pick-and-place operations) that the system has to perform, and on the environment (*e.g.*, structured, unknown, with humans) in which it operates. Moreover, relative performance depends heavily on the performance index (*e.g.*, energy consumption, task completion time, accuracy) that is considered. Thus, in the robot design process, many factors have to be weighed and designers inevitably have to deal with many trade-offs.

2.1.1 Geared Motors and Quasi Direct Drives

Among those actuation mechanisms that use DC motors, the two that are most often employed are Series Elastic Actuation (SEA) and Quasi-Direct Drive (QDD), which is simply a Geared Motor (GM) with reduced gear ratio [85], limited to roughly 10:1. In QDD actuation, the low-reduction transmission results in good transparency: low backlash, back-driveability, reduced reflected inertia of the motor, and lower friction (*i.e.*, higher power transmission efficiency). Moreover, QDD actuators have high control bandwidth, active compliance tuning capabilities, and good position controllability. Of course there are disadvantages in using QDD motors, such as low output torque and high Joule heating due to the necessity of working in high-current regimes [83–85].

2.1.2 Series Elastic Actuators

In an SEA, the motor is connected to a gearbox, which in turn is attached to one end of a spring, with the other end of the spring attached to the joint output. In terms of benefits, SEAs provide mechanically passive energy storage and regeneration, impact mitigation, high output torque, and increased peak output power. Moreover, an SEA can provide low mechanical output impedance, good force controllability, and safety in human-machine interactions. On the other hand, the drawbacks of SEAs include low control bandwidth and difficulty in controlling impulsive movements [78, 79, 84, 86].

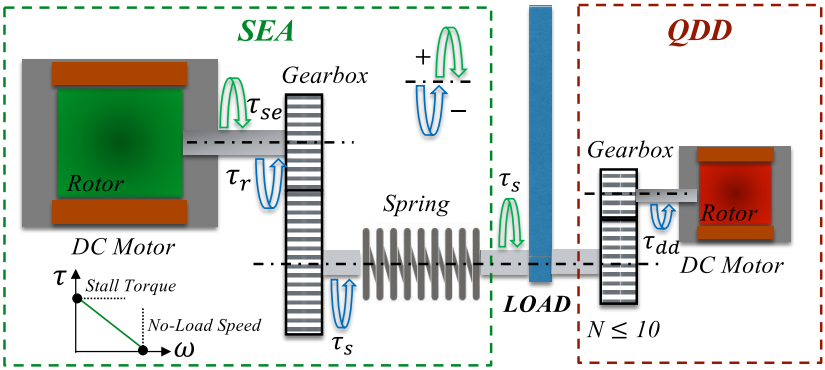


Figure 2.1: Schematic of the redundant actuation system: an SEA and a QDD work in parallel to actuate a single revolute joint

2.2 Redundant actuation system

Considering advantages and disadvantages of the two actuation mechanisms, there is a certain degree of complementarity between QDD and SEA. Recently, some researchers investigated the idea of exploiting the benefits of these two design approaches developing a high-bandwidth redundant actuator [83, 84] that uses QDD motors and SEA in parallel, as illustrated in Fig. 2.1. The results are promising, but it is still unclear whether this redundant actuation is energetically more efficient than SEA or QDD/GM alone.

To fairly consider the performance limitations of these systems, a framework is proposed to optimize design parameters using Co-Design, which simultaneously considers hardware and control in the design process. This means that hardware parameters (*e.g.*, spring stiffness, gear ratio) are included in an OCP as variables to be optimized, such that the final output is an actuation system that minimizes the energy consumed to perform a specific task.

This research starts from the analysis of the redundant actuation applied to a 1 degree of freedom (DoF) system and then extends the study to a 2-DoF manipulator. The analysis of the 1-DoF system compares the energy consumption of this actuation, considering a DD motor instead of a QDD for simplicity, with that of an SEA. In particular, considering a periodic sinusoidal motion with fixed amplitude, the variation of consumed energy as function of the oscillation frequency and other hardware

parameters is investigated.

Then, the same comparative analyses of energy consumption are carried out with the 2-DoF manipulator for two specific tasks: the classic swing-up problem, and a pick-and-place operation. In addition, following a co-design approach, some hardware parameters are included as decision variables of the optimization problem. This illustrates how Co-Design is a suitable way to design highly-efficient manipulators, considering both hardware and control.

2.2.1 1-DoF: OCP for determining minimal energy controls

As schematized in Fig. 2.1, the 1-DoF system with redundant actuation consists of a link connected to the ground by a revolute joint, which is actuated simultaneously by a DD motor and an SEA. The DD acts directly on the joint, while the SEA motor is connected to it by a gearbox and a torsional spring. The link dynamics is:

$$I_l\,\ddot{\theta}(t) = \tau_s(t) - m\,g\,\cos\left(\theta(t)\right)\,\frac{l}{2} + \tau_{dd}(t)\,, \tag{2.1}$$

where $\theta(t)$ is the link angle, I_l denotes its rotational inertia around the revolute joint, m is its mass, l its length and g is gravity. $\tau_s(t)$ and $\tau_{dd}(t)$ are respectively the output torque of the SEA spring and the DD motor. The case with only SEA is considered setting $\tau_{dd}(t) = 0$.

The dynamics of the SEA motor instead is:

$$I_{se}\,\ddot{\theta}_{se}(t) = \tau_{se}(t) - \tau_r(t)\,, \tag{2.2}$$

where $\theta_{se}(t)$ is the SEA motor angle, $\tau_{se}(t)$ is the torque generated by the SEA motor and $\tau_r(t)$ is the load torque acting on the SEA motor. For the 1-DoF case, the spring pre-load is set to zero. The dynamics of the SEA motor is coupled with that of the link by means of the SEA gearbox and spring:

$$\tau_r(t) = \frac{1}{N\eta} K_s \left(\frac{\theta_{se}(t)}{N} - \theta(t)\right)\,, \tag{2.3}$$

with K_s the SEA spring stiffness, N the gear ratio of the SEA gearbox, and η its efficiency [79], which accounts for the torque-dependent friction losses inside the gearbox. Note that $\tau_s(t) = N\eta\tau_r(t)$. For the sake of simplicity, the dependency of the efficiency on the gearbox loading is

neglected, as well as the fact that when the motor is not active ($\tau_{se}(t) = 0$) then the inefficiency would increase the braking capability. Indeed, modeling the gearbox efficiency as a factor multiplying the gear ratio makes the breaking torque zero when the motor is not active, which is not true for a real geared transmission subject to friction. To further simplify the analysis in the 1-DoF case, the inertias associated with the DD motor and with the SEA gearbox are not taken into account, assuming that they are dominated respectively by the inertia of the link (about four orders of magnitude lower) and that of the SEA motor (about one order of magnitude lower). Finally, in this simple 1-DoF case, only the thermal losses of the motors are considered, neglecting Coulomb and viscous friction.

The motion that is chosen for this analysis is a simple sinusoid with fixed amplitude $A = 5°$ and frequency f around the vertical position of the link, *i.e.*, $\theta(t) = \pi/2 + A\sin(ft)$. A small oscillation amplitude is chosen to limit the required motor torque at high frequencies. Enforcing this movement, the system of equations (2.1)-(2.3) is determined and can be solved analytically in the case with only the SEA, since the only control variable is the SEA motor torque. Therefore, if a solution exists it is unique.

Considering the redundant actuation instead, the system of equations becomes under-determined because of the DD motor torque. Thus, an OCP is formulated to realize the desired motion of the link with the least amount of energy:

$$\begin{aligned}
&\underset{x(\cdot),\,u(\cdot)}{\text{minimize}} && \Phi\left(x(\cdot), u(\cdot)\right) && \text{(2.4a)}\\
&\text{subject to} && \dot{x}(t) = f(t, x(t), u(t)) && \text{(2.4b)}\\
& && h(t, x(t), u(t)) \leq 0 && \text{(2.4c)}\\
& && g(t_f, x(0), x(t_f)) \leq 0 && \text{(2.4d)}
\end{aligned}$$

The state and control trajectories, with values $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$, are the decision variables. The objective function is represented by $\Phi(\cdot)$, while the dynamics, path and boundary constraints respectively by (2.4b), (2.4c) and (2.4d).

The cost function is the energy consumed to complete a cycle such that the time horizon is $t_f = 2\pi/f$. Any capability to regenerate energy from braking is initially neglected:

$$\Phi\left(\cdot\right) = \int_0^{t_f} \max(0, P_{se}(t)) + \max(0, P_{dd}(t))\ dt\,, \tag{2.5}$$

with P_{se} and P_{dd} the power associated respectively to the SEA and the DD motor, expressed as:

$$P_{se}(t) = \tau_{se}(t)\,\dot{\theta}_{se}(t) + \frac{\tau_{se}(t)^2}{K_m} \tag{2.6}$$

$$P_{dd}(t) = \tau_{dd}(t)\,\dot{\theta}(t) + \frac{\tau_{dd}(t)^2}{K_m}\,, \tag{2.7}$$

where K_m denotes the motor constant. The introduction of the $\max(\cdot)$ function would cause numerical problems because of its non-differentiability at zero. To avoid this non-smooth cost, the cost is reformulated by introducing two additional variables, $\epsilon_{se}(t)$ and $\epsilon_{dd}(t)$:

$$\Phi(\cdot) = \int_0^{t_f} \frac{P_{se}(t) + \epsilon_{se}(t)}{2} + \frac{P_{dd}(t) + \epsilon_{dd}(t)}{2}\; dt \tag{2.8}$$

while enforcing the constraints:

$$\epsilon_{se}(t) \geq -P_{se}(t), \quad \epsilon_{se}(t) \geq P_{se}(t) \tag{2.9}$$

$$\epsilon_{dd}(t) \geq -P_{dd}(t), \quad \epsilon_{dd}(t) \geq P_{dd}(t) \tag{2.10}$$

With this formulation, when $P < 0$ the solver optimizes to $\epsilon = -P$ so that $\frac{P+\epsilon}{2} = 0$. When instead $P \geq 0$ the solver optimizes to $\epsilon = P$ so that $\frac{P+\epsilon}{2} = P$.

Also the case of energy regeneration is considered, accounting for a battery that can be charged with 60% efficiency (as in the case of [87]):

$$\begin{aligned}\Phi(\cdot) = \int_0^{t_f} \Bigg[&\frac{P_{se}(t) + \epsilon_{se}(t)}{2} + \frac{P_{dd}(t) + \epsilon_{dd}(t)}{2} \\ &-0.6\left(\frac{\epsilon_{se}(t) - P_{se}(t)}{2} + \frac{\epsilon_{dd}(t) - P_{dd}(t)}{2}\right)\Bigg]\; dt\end{aligned} \tag{2.11}$$

The dynamic equations, (2.1)-(2.3), represent (2.4) in the OCP, and the constraint to track a desired angular acceleration $\ddot{\theta}(t) + f^2\,A\,\sin(f\,t) = 0$ is included in the path constraints (2.4c). The remaining path and boundary constraints (torque limits, initial and periodicity conditions), together with some implementation details are reported in the following list:

1. Remaining path constraints of the 1 DoF OCP:

$$\begin{aligned}-\tau_{max} \leq \tau_{se} \leq \tau_{max} \\ -\tau_{max} \leq \tau_{dd} \leq \tau_{max}\end{aligned} \tag{2.12}$$

2. Initial conditions of the 1 DoF OCP:

$$\theta(0) = \pi/2, \quad \dot{\theta}(t) = fA, \quad \Phi(0) = 0 \tag{2.13}$$

3. Periodicity conditions of the 1 DoF OCP:

$$\begin{aligned} \tau_{se}(0) &= \tau_{se}(t_f) \\ \tau_{dd}(0) &= \tau_{dd}(t_f) \\ \epsilon_{se}(0) &= \epsilon_{se}(t_f) \\ \epsilon_{dd}(0) &= \epsilon_{dd}(t_f) \\ P_{se}(0) &= P_{se}(t_f) \\ P_{dd}(0) &= P_{dd}(t_f) \end{aligned} \tag{2.14}$$

4. The accuracy of the model improved introducing a viscous friction term with constant friction coefficient β. It appears also in the SEA motor dynamics:

$$(I_{se} + I_g)\,\ddot{\theta}_{se}(t) = \tau_{se} - \frac{K_s}{\eta_{se}\,N_{se}}\left(\frac{\theta_{se}}{N_{se}} - \theta\right) - \beta\,\dot{\theta}_{se}(t) \tag{2.15}$$

5. The OCP (2.4) is specified using the optimization modeling language *Pyomo* [88] and solved using *IpOpt* [77] with the MA57 linear solver [89]. In order to express integral variables, as in the case of the OCP cost function, it is not taken advantage of the *Integral()* component that *Pyomo* provides with the framework *pyomo.dae* because it is still under development. An implicit definition is used instead, that is to declare the consumed energy as a simple variable, *i.e.* using the *Var()* component, and its derivative, the total power, with *DerivativeVar()*. Then the expression of the total power is provided as constraint. In this way the total energy consumed by the system from time 0 to t_f is obtained interrogating *Pyomo* about the value of the energy variable at time t_f.

2.2.2 2-DoF: OCP for Co-Design

For the 2-DoF case, a small gearbox is introduced on the DD motor because of the demanding torque requirements. Thus, what previously is called DD, is now referred to as Quasi-Direct Drive (QDD), since the gear ratio is limited to 10 [90].

The energy consumption analysis begins by considering the swing-up problem, illustrated in Fig. 2.2. It consists of making the manipulator lift a weight from the downward vertical configuration ($\theta_1(0) = -\pi/2,\ \theta_2(0) = 0$) up to the upward vertical one ($\theta_1(t_f) = \pi/2,\ \theta_2(t_f) = 0$). Thus, an OCP is formulated to minimize both the energy consumed and the time to complete this task. In an industrial context, task completion time and energy consumption both impact profits. The weights w_1 and w_2 are used in the cost function to set the relative importance of these two quantities. The choice of minimizing a single objective function, rather than performing a multi-objective optimization, is taken in light of the fact that both energy and time can be converted into money, using appropriate weights. Thus, carefully selecting the weights based on the type of company considered, the solution to our co-design problem represents the choice that minimizes the overall cost that the company has to bear for the design and control of such a robotic system achieving the task considered. In this 2-DoF case, some simplifying assumptions taken in the 1-DoF case are abandoned. Indeed, the inertias of gearboxes and QDD motors, as well as the viscous friction of joints and motors, are considered in the dynamic equations of the system.

First, the case with no energy regeneration is investigated, then the case with it. The co-design OCP is formulated as:

$$\begin{aligned}
&\underset{t_f, x(t), u(t), \rho}{\text{minimize}} && \Phi\left(t_f, x(t_f), u(t_f), \rho\right) && \text{(2.16a)}\\
&\text{subject to} && \dot{x}(t) = f(t, x(t), u(t), \rho) && \text{(2.16b)}\\
& && h(t, x(t), u(t), \rho) \leq 0 && \text{(2.16c)}\\
& && g(t_f, x(0), x(t_f/2), x(t_f), \rho) \leq 0 && \text{(2.16d)}
\end{aligned}$$

The key aspect making this problem one of Co-Design is the additional decision variables in ρ, which contains the design parameters: motor masses, spring stiffnesses and gear ratios. The motor constant and inertia of each motor are related to the motor mass by the relationships presented in [35]. For the sake of completeness, the following list reports the dynamic equations (2.16b), path (2.16c) and boundary constraints (2.16d) of the 2-DoF manipulator, as well as other implementation details:

1. The two degrees of freedom $\theta_1(t)$ and $\theta_2(t)$ are respectively the angle of the first link with respect to the horizontal and of the second link with respect to the first one. The redundant actuation

introduces four control variables represented by the torques $\tau_{se1}(t)$, $\tau_{se2}(t)$, $\tau_{gm1}(t)$ and $\tau_{gm2}(t)$ provided respectively by the two SEAs (characterized by gear ratios N_{se1} and N_{se2}, gearbox efficiencies η_{se1} and η_{se2} and spring stiffnesses K_{s1} and K_{s2}) and by the two GMs (with gear ratios N_{gm1} and N_{gm2} and gearbox efficiencies η_{gm1} and η_{gm2}). The gearbox efficiency is modeled as in the 1 DoF case. Mass, length and distance of center of mass from the joint have a subscript that specifies which link they refers to. I_1 and I_2 are not only the rotational inertias of the links around the joints, namely I_{l1} and I_{l2}. The first includes also the inertia of the GM (motor and gearbox inertias) on the first joint and the presence of the SEA and the GM on the second joint. The second one takes into account also the inertia of the GM on the second joint. They are expressed as follows:

$$I_1 = I_{l1} + (m_{se2} + m_{gm2} + 2m_g)\, l_1^2 + (I_{gm1} + I_g)\, N_{gm1}^2 \tag{2.17}$$

$$I_2 = I_{l2} + (I_{gm2} + I_g)\, N_{gm2}^2 \tag{2.18}$$

Where m_{se2} and m_{gm2} are the masses of the SEA and GM motors on the second joint, while m_g and I_g are mass and inertia of the gearboxes. m_g and I_g are considered as constants and not dependent on the gear ratio because it is observed that their real values do not vary much for the selected range of gear ratios [91].

2. Dynamics of the 2 DoF system (in addition to (2.15) for the SEAs):

$$\begin{aligned}
&\left(m_{l1}\, l_{cm1}^2 + m_{l2}(l_1^2 + l_{cm2}^2 + 2\, l_1\, l_{cm2}\, cos(\theta_2)) + I_1 + I_2\right) \ddot{\theta}_1 + \\
&(m_{l2}\left(l_{cm2}^2 + l_1\, l_{cm2}\, cos(\theta_2)\right) + I_2)\ddot{\theta}_2 - \\
&\left(m_{l2}\, l_1\, l_{cm2}\, sin(\theta_2)\dot{\theta}_2^2 + 2\, m_{l2}\, l_1\, l_{cm2}\, sin(\theta_2)\, \dot{\theta}_1\, \dot{\theta}_2\right) + \\
&g\, cos(\theta_1)(m_{l1}\, l_{cm1} + m_{l2}\, l_1) + m_{l2}\, l_{cm2}\, g\, cos(\theta_1 + \theta_2) - \\
&K_{s1}\left(\frac{\theta_{se1}}{N_{se1}} - \theta_1\right) - \left(\tau_{gm1} - \beta\, N_{gm1}\dot{\theta}_1\right) \eta_{gm1}\, N_{gm1} = 0
\end{aligned} \tag{2.19}$$

$$\begin{aligned}
&\left(m_{l2}\, l_{cm2}^2 + I_2\right) \ddot{\theta}_2 + \left(m_{l2}\left(l_{cm2}^2 + l_1\, l_{cm2}\, cos(\theta_2)\right) + I_2\right) \ddot{\theta}_1 + \\
&\left(m_{l2}\, l_1\, l_{cm2}\, sin(\theta_2)\dot{\theta}_1^2\right) + (m_{l2}\, l_{cm2}\, g\, cos(\theta_1 + \theta_2)) - \\
&K_{s2}\left(\frac{\theta_{se2}}{N_{se2}} - \theta_2\right) - \left(\tau_{gm2} - \beta\, N_{gm2}\dot{\theta}_2\right) \eta_{gm2}\, N_{gm2} = 0
\end{aligned} \tag{2.20}$$

Similarly to the 1 DoF model, to consider the case of only SEAs, the last terms in (2.19)- (2.20) involving the GMs should be neglected.

3. Path constraints of the 2 DoF OCP:

$$\begin{aligned}
&\eta_k - N_k^{-0.0952} = 0 \qquad k = 1 \dots 4 \\
&P_{se,k} - \tau_{se,k}\,\dot{\theta}_{se,k} - \frac{\tau_{se,k}^2}{K_{m_{se,k}}} = 0 \qquad k = 1, 2 \\
&P_{gm,k} - \tau_{gm,k}\,\dot{\theta}_k\,N_{gm,k} - \frac{\tau_{gm,k}^2}{K_{m_{gm,k}}} = 0 \qquad k = 1, 2 \\
&K_{m_{se,k}} - 0.0567\; m_{se,k}^{1.8} = 0 \qquad k = 1 \dots 2 \\
&K_{m_{gm,k}} - 0.0567\; m_{gm,k}^{1.8} = 0 \qquad k = 1 \dots 2 \\
&I_{se,k} - 2.85\,10^{-5}\; m_{se,k}^{1.72} = 0 \qquad k = 1 \dots 2 \\
&I_{gm,k} - 2.85\,10^{-5}\; m_{gm,k}^{1.72} = 0 \qquad k = 1 \dots 2
\end{aligned} \tag{2.21}$$

4. Boundary constraints of the 2 DoF OCP for the swing up task:

$$\begin{aligned}
&\theta_1(0) = -\theta_1(t_f) = -\pi/2 \\
&\theta_2(0) = \theta_2(t_f) = 0
\end{aligned} \tag{2.22}$$

5. Boundary constraints of the 2 DoF OCP for the pick-and-place task:

$$\begin{aligned}
&\theta_1(0) \geq 0, \quad \Phi(0, \rho) = 0, \quad \theta_1(t_f) \geq 0 \\
&l_1\,cos(\theta_1(0)) + l_2\,cos(\theta_1(0) + \theta_2(0)) = \frac{l_1}{2} \\
&l_1\,sin(\theta_1(0)) + l_2\,sin(\theta_1(0) + \theta 2(0)) = 0 \\
&l_1\,cos(\theta_1(t_f/2)) + l_2\,cos(\theta_1(t_f/2) + \theta_2(t_f/2)) = \frac{3l_1}{2} \\
&l_1\,sin(\theta_1(t_f/2)) + l_2\,sin(\theta_1(t_f/2) + \theta_2(t_f/2)) = l_1 \\
&l_1\,cos(\theta_1(t_f)) + l_2\,cos(\theta_1(t_f) + \theta_2(t_f)) = \frac{l_1}{2} \\
&l_1\,sin(\theta_1(t_f)) + l_2\,sin(\theta_1(t_f) + \theta_2(t_f)) = 0 \\
&\theta_{se1}(0) = \theta_{se1}(t_f), \quad \theta_{se2}(0) = \theta_{se2}(t_f)
\end{aligned} \tag{2.23}$$

6. Due to the non-convexity of the OCP in the case of the 2 DoF system, in many simulations *IpOpt* failed to find a local minimum.

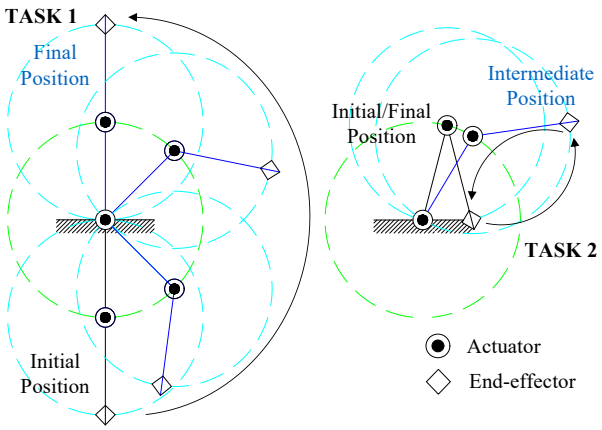


Figure 2.2: Swing-up task and pick-and-place operation performed by the 2-DoF system. The green and blue circles represent instantaneous position domains for the first and second link, respectively

The capability of *IpOpt* to find a solution is enhanced and the time of the simulations is reduced by replacing the standard linear solver with the MA57 linear solver [89].

Without energy regeneration, the expression of the cost function (2.16a) is:

$$\Phi(\cdot) = w_1 \int_0^{t_f} \left(\sum_{k=1}^{2} \frac{P_{se,k}(t,\rho) + \epsilon_{se,k}(t,\rho)}{2} + \right. \\ \left. + \sum_{k=1}^{2} \frac{P_{dd,k}(t,\rho) + \epsilon_{dd,k}(t,\rho)}{2} \right) dt + w_2 t_f^2 \tag{2.24}$$

To avoid trivial solutions that would have exploited only the torque given by the pre-load of the SEAs springs, for this task the pre-load is set to zero.

The effects of this redundant actuation are also studied considering a pick-and-place operation for a 2-DoF manipulator (see Fig. 2.2). The task consists of moving the manipulator from an initial configuration to a certain position of the end-effector, and then bringing it back to the initial configuration. Specifically, the initial/final configuration is given by the end-effector being at the same height of the first joint and at

a distance of $l_1/2$ from it, while in the intermediate configuration the end-effector reaches a height of l_1 and a distance of $3l_1/2$.

For this task, the springs pre-load are let remain free, but added a constraint that the final pre-load must be equal to the initial one.

2.3 Results

This section reports the results for the 1-DoF and 2-DoF models. First, the energy consumption of the redundant actuator and SEA for the 1-DoF system is analyzed. It turns out that for sinusoidal motions the energy saving using the redundant actuation can exceed 90%. Moreover, the resulting optimal control strategy is very similar to the latching control used in energy harvesters [92, 93], which consists in locking the moving body of a heaving buoy device at the end of the oscillation, so when its velocity is zero, and then releasing it when its velocity is back in phase with the wave excitation force.

With the 2-DoF system, the comparison extends also to GMs (*i.e.*, DC motors attached to gearboxes with gear ratios up to $N = 200$), redundant actuation at both the joints, and redundant actuation at the first joint and SEA at the second joint. The 2-DoF system carried out two different tasks: swing-up and pick-and-place. A co-design framework is built to find optimal hardware parameters and control trajectories for each task. Our analysis shows that, even though latching-like control is not optimal for non-sinusoidal motions, the redundant actuation can still be more efficient than SEAs and GMs.

In addition, also the closed-loop behavior of the actuators under perturbations is studied. It turns out that tracking the desired trajectories while maintaining low energy consumption can sometimes be facilitated by the actuator redundancy.

2.3.1 1-DoF: Test Details

The test with the 1-DoF system consists of fixing the motion of the joint (sinusoid) and optimizing the redundant actuation control to minimize the energy consumption. In the case with only the SEA, the motor torque is analytically determined. The frequency range in the analysis goes from 0.1 to 10 Hz, with 0.1 Hz resolution. The set of values for the spring stiffness is $[30, 50, 100]$ Nm/rad, while for the gear ratio of the SEA it is $[5, 10, 15, 20, 30, 50, 100, 150, 200]$.

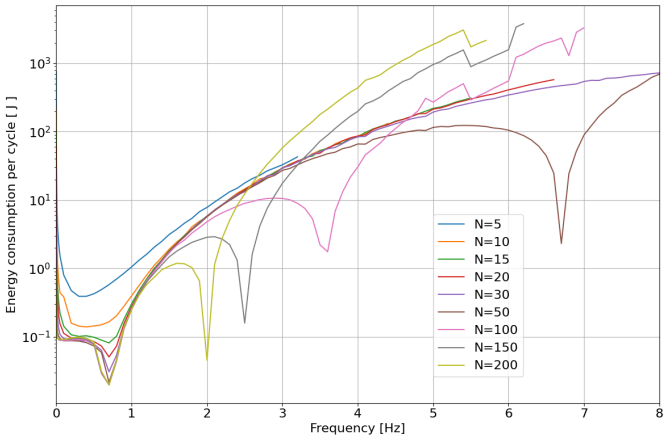


Figure 2.3: Energy use as a function of the oscillation frequency for the SEA with spring constant $K_s = 30\,\mathrm{Nm/rad}$ and 9 different gear ratios N.

The same brushless DC motor model is considered for both the SEA and the DD motor. Motor specifications include: peak torque $\tau_{max} = 7.13\,\mathrm{Nm}$, rotor inertia $I_{se} = 10^{-5}\,\mathrm{kg \cdot m^2}$ and motor constant $K_m = 0.64\,\mathrm{Nm}/\sqrt{\mathrm{Watt}}$. The gearbox efficiency is considered to scale exponentially with the gear ratio as suggested in [35].

For all the possible combinations of oscillation frequency, spring stiffness and gear ratio, the OCP (2.4) is solved after transcription using direct collocation with a Lagrange-Radau scheme, 30 finite elements, and 3 collocation points per element.

2.3.2 1-DoF: Test Results

Our results show that the redundant actuation is energetically more efficient than the SEA. Fig. 2.3 shows the energy consumption of the SEA, considering a 30 Nm/rad spring stiffness and different gear ratios, while Fig. 2.4 shows the case with redundant actuation. Both cases assumed no energy regeneration. The valleys in Fig. 2.3 occur at the system natural frequencies, which vary depending on the gear ratio. Indeed, the higher the gear ratio, the lower the system natural frequency. The natural frequency also depends also on the spring stiffness: fixing the gear

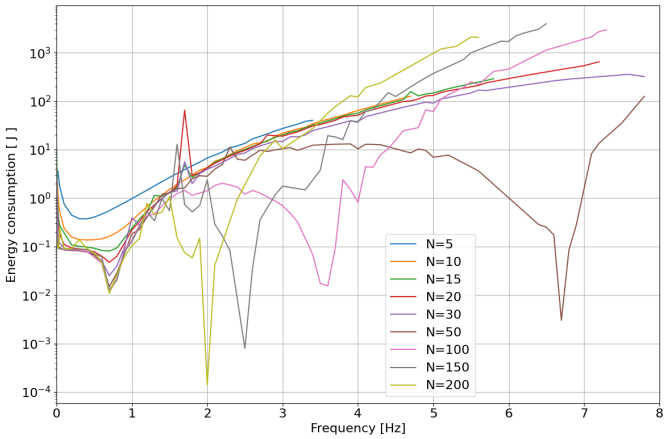


Figure 2.4: Energy use as a function of the oscillation frequency for the redundant actuator with spring constant $K_s = 30\,\mathrm{Nm/rad}$ and 9 different gear ratios N.

ratio, the natural frequency shifts to higher values as the spring stiffness increases.

To better understand the energy advantage of the redundant actuation, Fig. 2.5 shows the energy savings achieved with the redundant actuation in absolute terms while Fig. 2.6 illustrates the energy savings as a percentage of the SEA energy consumption. To improve readability, the data of Fig. 2.5 and Fig. 2.6 are filtered with a digital Butterworth filter of order 1 with critical frequency equal to 0.2 half-cycles/s (sampling frequency = 2 half-cycles/s). The energy savings can be very large, up to 99%, depending on the oscillation frequency, the gear ratio and the spring stiffness. For instance, in the range 3-4 Hz with $N = 100$ and $K_s = 30\,\mathrm{Nm/rad}$, the redundant actuation can save more than 90% of energy compared to the SEA, which means saving about 10 J. In the same frequency range, if one considers $N = 200$ then the relative energy saving decreases to about 70% but the absolute value of energy saving increases approximately to 100 J. On the other hand, at 1 Hz the energy saving is much lower: about 40% for high gear ratios ($N \geq 100$), and less than 20% for low gear ratios ($N \leq 20$).

Insight into the large energy savings achievable with the redundant actuation comes from how each actuator produces the desired joint

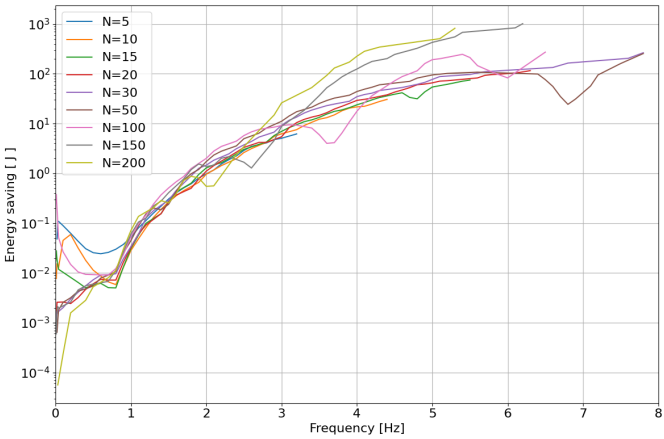


Figure 2.5: Energy saving of the redundant actuation expressed in absolute values.

torque. With only the SEA, the torque is provided only by the spring, while with the redundant actuation it comes also from the DD motor. The spring torque mainly stems from the motion of the SEA rotor being opposite to that of the joint. Since in this test the joint trajectory is fixed, the SEA rotor angle completely defines the spring torque. Thus, with only the SEA, the analytical solution for the rotor angle trajectory can be computed. To make the rotor achieve this motion, the motor must provide a sinusoidal torque (opposite to the spring torque which is seen as load torque by the SEA). For instance, Fig. 2.7 compares the torque, angular velocity, and power from the SEA and the load for $N = 100$, $Ks = 30\,\mathrm{Nm/rad}$ and $f = 3\,\mathrm{Hz}$, which led to energy savings of more than 90%. As the figure shows, the SEA motor must be always powered, which implies a high energy consumption due to the SEA motor velocity reaching very high values (up to $1500\,\mathrm{rad/s}$). In two large time intervals the SEA positive power (due to no energy regeneration) is greater than zero and reaches almost $100\,\mathrm{W}$.

With the redundant actuation instead, the SEA motor is powered only during two short intervals ($0.07 - 0.1\,\mathrm{s}$ and $0.22 - 0.26\,\mathrm{s}$), when it almost halts the motion of its rotor (see Fig. 2.8). The SEA rotor is not completely blocked so that the thermally dissipated power is compensated by negative mechanical power, resulting in nearly always

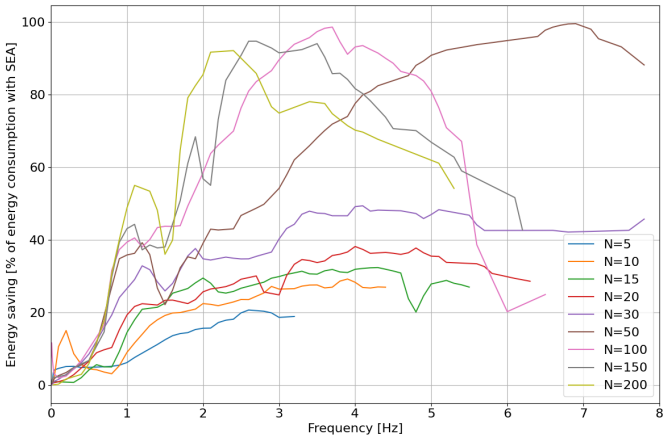


Figure 2.6: Energy saving of the redundant actuation expressed as percentage of SEA energy consumption.

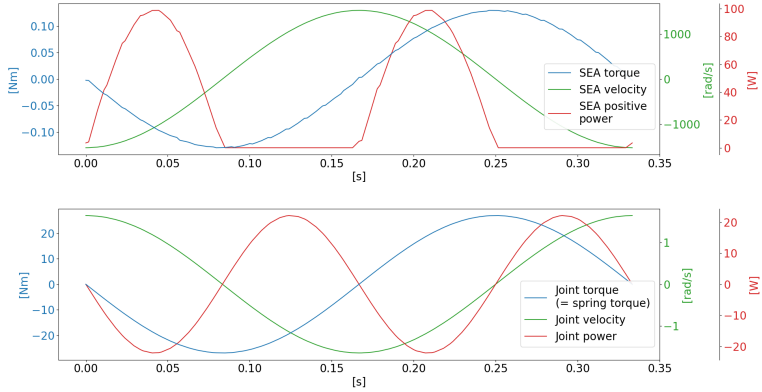


Figure 2.7: SEA: torque, velocity and power of (top) motor and (bottom) joint required to perform a 3Hz sinusoidal motion. The SEA positive power is the maximum between zero and the power consumed by the SEA (no energy regeneration case).

non-positive total power. It is interesting to notice that the solver found a control strategy very similar to the *latching control*, which is implemented in many energy harvesters to maximize energy generation, by

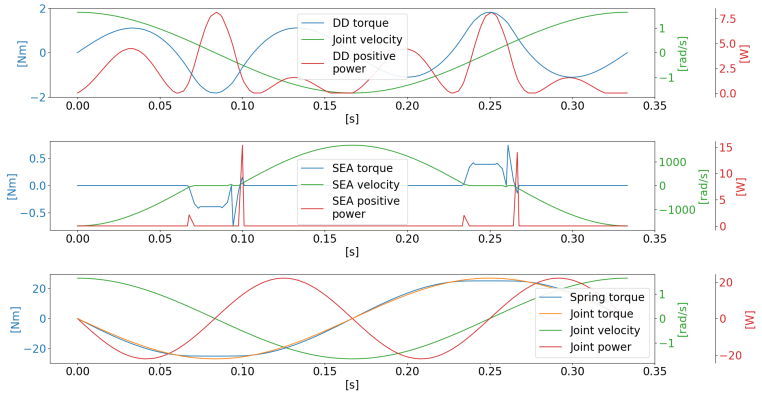


Figure 2.8: Redundant actuation: torque, velocity and power of motors and joint required to perform a 3Hz sinusoidal motion. The DD and SEA positive power is the maximum between zero and the power consumed respectively by the DD and the SEA (no energy regeneration case).

keeping the SEA rotor velocity in counter-phase with the joint velocity [92,93]. Indeed, in heaving buoy wave energy converters, the oscillating body is often latched when its velocity vanishes in order to keep the latter in phase with the wave excitation force. This condition is proved to be required for the maximization of the energy generation [94]. In the redundant actuation, this strategy is applicable thanks to the DD motor, which provides the additional torque needed to perform correctly the sinusoidal motion. Since this torque is rather small, as well as the joint velocity, the overall energy consumption remains small (peak positive power less than 8 W).

2.3.3 2-DoF: Test Details

Four actuation architectures are considered: only GMs, only SEAs, redundant actuation on both joints (Full redundant) and redundant actuation on the first joint with SEA on the second one (redundant+SEA). For both tasks, the weights w_1 (associated to energy) and w_2 (associated to time) in the cost function (2.24) are set to 1 and 0.1, respectively. Moreover, to improve accuracy, the number of finite elements are increased to 100. Finally, the case with 60% energy regeneration is also investigated.

Since the solution that *IpOpt* could find is strongly dependent on the

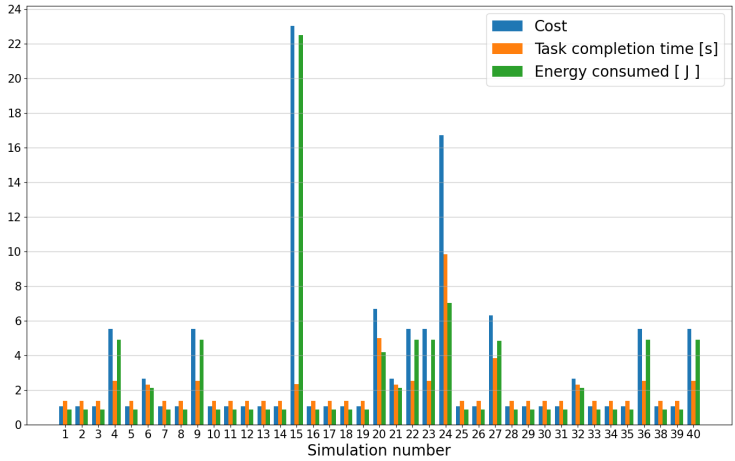


Figure 2.9: Results of 40 simulations considering the pick-and-place task and the actuation architecture with only SEAs. Because of the non-convexity of the problem, the solution strongly depends on how the decision variables are initialized.

initial guess of the hardware design parameters, the initialization values are randomized and the simulations are repeated 40 times. Beyond 40 random initial guesses, the change in the best solution is found to be negligible. To highlight the non-convexity of this co-design problem, which explains the dependency of the solution on the initial guesses, Fig. 2.9 illustrates the results of 40 simulations considering the pick-and-place task and the actuation architecture with only SEAs. In this case, 65% of simulations lead to the same minimum-cost solution.

2.3.4 2-DoF: Test Results - Swing-Up

The following results concern a swing-up task with no load. Tab. 2.1 summarizes the results with no energy regeneration and compares the energy consumption and task completion time of the different actuation architectures. The Full redundant actuation turned out to be the least effective, in terms of both energy consumption and task completion time. Instead, the redundant+SEA configuration performed the best. It allowed for a 5% energy saving compared to only SEAs, with a

Table 2.1: Task Completion Time and Energy Consumption with Different Actuation Architectures

Architecture	Task Time [s]		Energy [J]		Cost function	
	Swing Up	***Pick &Place***	***Swing Up***	***Pick &Place***	***Swing Up***	***Pick &Place***
GMs	5.71	3.38	226.68	42.57	229.94	43.72
SEAs	4.17	1.36	225.17	0.86	226.90	1.04
Full redundant	10.89	1.59	264.72	1.16	276.59	1.42
Redundant + SEA	5.25	1.37	214.01	0.80	216.77	0.99

Table 2.2: Co-design Results for Swing Up Task

Architecture	Motor Mass [kg]		Gear Ratio		Spring Stiffness
	GM	***QDD***	***GM***	***QDD***	***[N·m/rad]***
1^{st} GM	10.00	—	18.79	—	—
2^{nd} GM	0.10	—	1.00	—	—
1^{st} SEA	10.00	—	21.25	—	139.12
2^{nd} SEA	0.10	—	200.00	—	17.21
1^{st} Redundant	10.00	10.00	8.51	8.47	43.61
2^{nd} Redundant	0.10	0.10	200.00	1.00	74.88
1^{st} Redundant	10.00	0.10	13.35	10.00	70.65
2^{nd} SEA	0.10	—	200.00	—	24.75

26% increase in task completion time. In contrast, when compared to only GMs, around the same energy saving comes accompanied by an 8% reduction in completion time. Tab. 2.1 reports also the objective function values, supporting the fact that the redundant+SEA configuration seems to be the optimal choice for the swing-up problem.

Tab. 2.2 lists the optimal hardware parameters found from solution of the OCP (2.16a) using a co-design framework. In all cases, the mass of the motors on the first joint is equal to its upper bound (10 kg), while for the motors on the second joint the optimal mass corresponds to the lower bound (0.1 kg). This is reasonable: the inertia seen at the first joint would increase if the motors on the second joint are heavier, while the motors actuating the first joint are located at the robot base and

thus do not increase the inertia. The optimal gear ratios on the second joint in both cases with GMs and with Full redundant actuation are at the lower bound, namely 1. Therefore, the GM and QDD at the second joint contribute very little to the motion, while considerably increasing the inertia of the system; for this reason the solver sets also the motor masses to the lower bound.

This finding motivated us to investigate the design with redundant actuation at the first joint and only SEA at the second one. That the optimal gear ratios of the SEAs on the second joint are at their upper bound means that the solver tries to maximize the contribution of these SEAs without increasing the inertia of the system. However, if the gearbox mass is modelled as function of the gear ratio, then the optimal gear ratios would be lower than 200 because they would affect directly the system inertia and so the energy use.

2.3.5 2-DoF: Test Results - Pick and Place

Tab. 2.1 summarizes the results of the pick-and-place operation. The three actuation systems that employ SEAs clearly outperform the one with GMs. This is due to the task periodicity, which enables exploiting the springs pre-load to perform the motion. As observed with the swing-up task, also in this case the best results are achieved using the redundant actuation on the first joint and SEA on the second one. Compared to the Full redundant actuation, the energy savings is 31% and the completion time is reduced by 13.8%. Considering the case with only SEAs instead, the task can be performed taking almost the same time but consuming 7% less energy.

Tab. 2.3 presents the resulting optimal hardware parameters for the pick-and-place task. As observed with the swing-up task, in all cases the optimal mass of the motors on the first joint is the maximum allowed, as it happens also for the gear ratios of the SEAs on the second joint, while the optimal motor mass of the QDD on the second joint of the Full redundant actuation as well as its gear ratio hit the lower bounds. The fact that the values of gear ratio of the SEAs on the first joint are equal to their upper bound may be due to the high-torque demanding static conditions set for this task in the initial-intermediate-final configurations. With this task, it is not convenient to adopt the strategy of using the SEAs to store and release mechanical power through the swinging motion of the links, as in the case of the swing-up task. This may explain why the solver sets the spring stiffnesses of the SEAs on the first joint to the

Table 2.3: Co-design Results for Pick&Place Task

Architecture	Motor Mass [kg]		Gear Ratio		Spring Stiffness
	GM	***QDD***	***GM***	***QDD***	***[N·m/rad]***
1^{st} GM	10.00	—	89.41	—	—
2^{nd} GM	4.34	—	10.71	—	—
1^{st} SEA	10.00	—	200.00	—	250.00
2^{nd} SEA	3.17	—	200.00	—	9.23
1^{st} Redundant	10.00	10.00	200.00	5.66	250.00
2^{nd} Redundant	2.75	0.10	200.0	1.00	5.51
1^{st} Redundant	10.00	10.00	200.00	4.52	250.00
2^{nd} SEA	3.18	—	200.00	—	9.18

maximum value, so as to increase the bandwidth of the SEAs.

As presented so far, the co-design results for both tasks show that the hardware design parameters hit the upper or lower bounds many times. Whereas in some cases this phenomenon can be explained quite easily, in other cases finding an intuitive explanation is much more complicated and the results may actually be misleading. A solution to this problem could be the use of regularization terms in the cost function that prevent those hardware parameters from hitting their upper and lower bounds.

The results up to this point suggest that a periodic task can be efficiently achieved in ideal settings using SEAs, thanks to their capability to store and release mechanical power without wasting energy. The benefits of adding a DD in parallel to the SEA in the 1-DoF system, that allows for a latching control strategy, are observed only to a small extent with the 2-DoF system and the tasks considered. Nonetheless, choosing other tasks may highlight more effectively the energy efficiency of the redundant actuation.

2.3.6 2-DoF: Feedback Control

The previous subsections investigated the energy efficiency of different actuation architectures (SEA, GM, Full redundant, redundant+SEA) in ideal settings, showing that the redundant actuator and redundant+SEA can lead to energy savings. However, this does not suffice to claim that this actuator could perform well in the real world. For this reason, the behavior of different actuators is now analyzed in more realistic settings, in which the system has to cope with modeling errors and disturbances

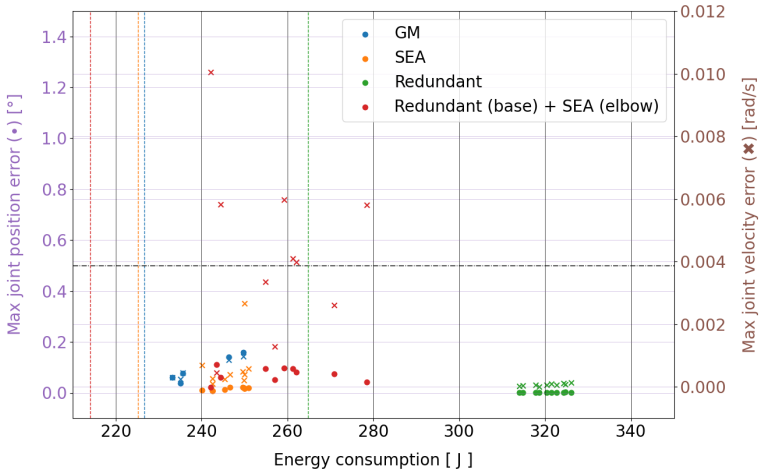


Figure 2.10: Swing-Up Task: energy consumption and maximum final joint-state error with PD control and disturbances in the optimized hardware parameters as well as impulsive variations of joint accelerations. The dots and crosses mark the position and velocity error against energy, respectively.

using feedback control.

To this end, a PD controller with hand-tuned gains is used to observe how energy consumption and task completion accuracy varied due to modeling errors in the hardware parameters and joint acceleration disturbances. 10 simulations are carried out for each actuation system, randomly selecting the magnitude of the disturbances up to 1% of the nominal value of the optimized hardware parameters and considering impulsive variations of the joint accelerations ($10\,\mathrm{rad/s^2}$) randomly occurring between 25% and 75% of the task completion time. Figs. 2.11 and 2.10 show the energy consumption and the maximum joint state error for the pick-and-place operation and swing-up task, respectively. The error is measured at the intermediate and final state for the pick-and-place task and only at the final state for the swing-up task. The dashed vertical lines represent the energy consumption in absence of any disturbance, while the dashed horizontal line is a subjective threshold representing the maximum position error ($0.5°$) below which the task is considered to be successful. Since nominal energy consumption with only SEAs and with redundant+SEA are very similar (respectively 0.86

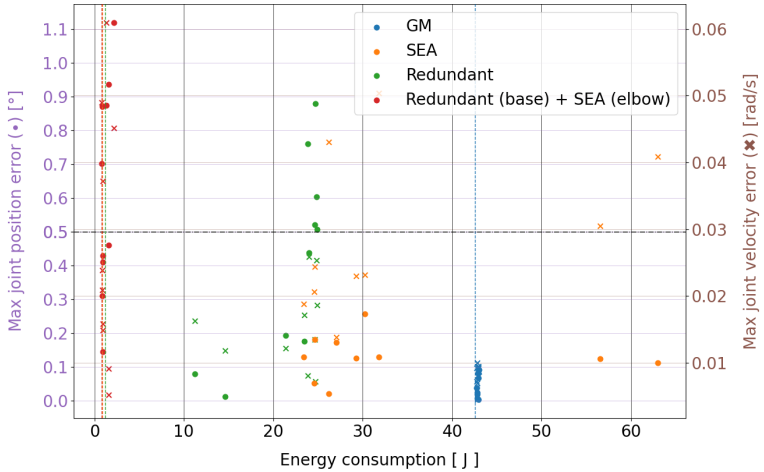


Figure 2.11: Pick-and-Place Task: energy consumption and maximum between intermediate and final joint-state error with PD control and disturbances in the optimized hardware parameters as well as impulsive variations of joint accelerations. The dots and crosses mark the position and velocity error against energy, respectively.

J and 0.80 J), the corresponding lines overlap.

In the pick-and-place task, the open-loop behavior of the system with redundant+SEA also extends to closed-loop with disturbances: the energy consumption is very close to the nominal one, although the maximum joint position error exceeds the threshold in half of the simulations. The GM actuation also ensures an energy consumption very close to the nominal one, in addition to an overall high accuracy and precision. The other two actuation systems instead perform poorly both in terms of actual energy consumption and task accuracy. Overall, the maximum position and velocity errors considering the intermediate and final configurations remain consistently bounded.

The results are different for the swing-up task: SEA and redundant+SEA perform similarly, while good accuracy is obtained with Full redundant actuation to the detriment of a high increment in energy consumption. In Fig. 2.10, the results of only 5 simulations with GMs are shown because in half of the simulations the controller failed to stabilize the system. In terms of accuracy compared to the pick-and-place task, the results for the swing-up task may be explained by the insta-

bility of the desired final configuration. An exception is represented by the Full redundant actuation, which may benefit from the combined action of QDD and SEA on the second joint to better reject acceleration disturbances.

2.4 Robustness reasoning

This section briefly describes how to modify a co-design framework as the one presented in this chapter to add robustness reasoning and find a pair of optimized hardware and nominal control trajectories that sustain their efficiency when stabilized via feedback control. This framework is the result of the collaborative work led by G. Bravo-Palacios at the University of Notre Dame, which involved also the author of this thesis and was presented in a recent publication [54]. Thus, this section merely outlines the methodology and the key findings. Although the results reported in [54] concern robotic systems that are different from the redundant actuation system considered in this chapter, its applicability to such robotic system would be straightforward.

As highlighted in [95], if a nominal trajectory is computed by considering factors such as modeling uncertainties, unplanned external forces, or state estimation errors, the control effort to track such a trajectory should decrease. Following the same intuition, this effect may be even more pronounced if the hardware is co-optimized with the nominal trajectory and controller. For this reason, the proposed co-design framework variant is based on the premise that the solution of the classic co-design problem can be made more robust by incorporating feedback design in the optimization process. This framework utilizes a probabilistic model to make design decisions by accounting for potential perturbations to a nominal scenario. These perturbations may entail alterations in the environment's geometry (e.g., changes in terrain inclination angle or height) or the application of external forces to the robot during a task. Stochastic Programming (SP) is used to model uncertainty and extend the co-design problem to account for multiple scenarios. The results demonstrate the benefits of co-optimizing the hardware, nominal trajectory, and feedback controller in terms of robustness to external disturbances, as well as the fact that the hardware selection significantly influences the robot's control actions, overall performance, and energy efficiency.

2.4.1 Stochastic Programming (SP) formulation

This subsection outlines the formulation of the robust variant of a co-design framework, as the one presented in Chapter 2. The framework bases its design decisions on probabilistic models of possible perturbations ω_{ξ_i} (*e.g.*, random push or change in terrain's properties) to a nominal scenario ξ^*. By combining a nominal scenario with a perturbation, the framework generates a disturbed scenario $\xi_i = \xi^* + \omega_{\xi_i}$. Considering the SP syntax [96], the co-design problem is expanded from (2.16a)-(2.16d) to encompass multiple scenarios via a two-stage SP problem. This problem is denoted as $\mathcal{P}(\{\omega_{\xi_i}\}_{i=1}^{N_s})$, indicating the disturbance scenarios ξ_i that are taken into consideration. In SP context, the term "stage" refers to an abstract phase of construction of a mathematical program. The first stage encompasses decisions related to nominal state and control trajectories, as well as hardware and feedback control parameters. These first-stage decisions are consistent across all perturbed scenarios, which are addressed in the second stage. Conversely, second-stage decisions are related to each perturbed scenario. It is important to note that passing through each stage does not mean solving a distinct problem but entails incorporating variables and constraints to create a unified optimization problem. Thus, the problem formulation is as follows:

$$\mathcal{P}(\{\omega_{\xi_i}\}_{i=1}^{N_s}) \doteq \tag{2.25}$$

$$\begin{aligned}
\underset{t_f,\gamma^*(\cdot),\{\gamma_{\xi_i}(\cdot)\},\rho,\mu(\cdot)}{\text{minimize}} \quad & \Phi_{\xi^*}\left(\gamma^*(\cdot),\rho\right) + \sum_{i=1}^{N_s} p(\xi_i)\Phi_{\xi_i}\left(\gamma_{\xi_i}(\cdot),\rho\right) && \text{(2.26a)}\\
\text{subject to} \quad & \dot{x}_{\xi^*}(t) = f(t,\gamma^*(t),\rho) && \text{(2.26b)}\\
& h_{\xi^*}(t,\gamma^*(t),\rho) \leq 0 && \text{(2.26c)}\\
& g(t_f,\gamma^*(0),\gamma^*(t_f),\rho) \leq 0 && \text{(2.26d)}\\
& \dot{x}_{\xi_i}(t) = f(t,\gamma_{\xi_i}(t),\omega\xi_i(t),\rho) && \text{(2.26e)}\\
& h_{\xi_i}(t,\gamma_{\xi_i}(t),\omega\xi_i(t),\rho) \leq 0 && \text{(2.26f)}\\
& g(t_f,\gamma_{\xi_i}(0),\gamma_{\xi_i}(t_f),\omega\xi_i(t_i),\omega\xi_i(t_f),\rho) \leq 0 && \text{(2.26g)}\\
& u_{\xi_i}(t) = \mu(x_{\xi_i}(t),\gamma^*(t)) && \text{(2.26h)}
\end{aligned}$$

where $\gamma(t) = (x(t), u(t))$ is the state-control trajectory, (2.26e)-(2.26g) are respectively the dynamics, path and boundary constraints for the perturbed scenarios $\xi_i,\ i \in \{1, ..., N_s\}$, (2.26h) indicates a feedback control policy $\mu(\cdot)$, and $p(\xi_i) \in [0, 1]$ is the probability of occurrence of

Algorithm 2: Robust Co-Design

Inputs: $\gamma_r^0,\ \rho_r^0,\ K_r^0,\ N_s^0$
$k \leftarrow 0$
repeat
  $\{\gamma^{*k}, \rho^{*k}, K^k\} = \text{solve}(\mathcal{P}(\{\omega_{\xi_i}\}_{i=1}^{N_s^k}),\ \gamma_r^k,\ \rho_r^k,\ K_r^k)$
  $\gamma_r^{k+1} \leftarrow \gamma^{*k}$
  $\rho_r^{k+1} \leftarrow \rho^k$
  $K_r^{k+1} \leftarrow K^k$
  $N_s^{k+1} \leftarrow N_s^k + n_s$
  $k \leftarrow k + 1$
until *stopping criterion is satisfied*;

the scenario ξ_i. The feedback control policy is in the form of a linear feedback:

$$\mu(x(t), \gamma^*(t)) = u^*(t) - K(t)(x(t) - x^*(t)) \tag{2.27}$$

and it is optimized by including the time-varying gain matrix $K(t)$ as a decision variable in the optimization process.

2.4.2 Robust co-design algorithm

As presented in [54], to exploit effectively the SP formulation (2.26) and to deal with scalability, one can follow a homotopy approach that iteratively refines a warm-start solution $\gamma_r(\cdot), \rho_r$ and $K_r(\cdot)$ as additional scenarios are taken into account. This strategy is illustrated in Algorithm 2. At each iteration of the algorithm, n_s new scenarios are added to the co-design problem $\mathcal{P}(\{\omega_{\xi_i}\}_{i=1}^{N_s})$, which is solved via TO to compute $\gamma^*, \rho^*, K(\cdot)$ that are used to warm-start the co-design problem at the next iteration. The stopping criterion can be represented by not observing changes in the design variables or when the co-design problem becomes not feasible.

2.4.3 Results

The experiments reported in [54] involve three systems, namely a 7-DoF manipulator actuated by gear motors, a 3-link manipulator with SEAs, and a 5-DoF planar monopod robot. The first case study, in which the manipulator has to lift a load while being perturbed by external forces

applied along its last link, shows that the co-optimized nominal trajectory and feedback controller led to the smallest tracking error for the considered scenarios, with the co-optimized nominal trajectory requiring less energy than the fixed one. Considering 10 scenarios used during the optimization, the robust co-design allowed to reduce the tracking error of 91% and the energy consumed of 5.4% with respect to the decoupled nominal co-design controlled via LQR. The second case study, which considers the same task as the previous test, highlighted even more the importance of coupling hardware, nominal trajectory and feedback design: LQR was not able to make the manipulator with decoupled co-optimized design track the reference trajectory when disturbances are applied, whereas the co-optimized controller was able to do so. Finally, the last case study involving a planar monopod robot that has to jump over a stair obstacle whose height is varied in the perturbed scenarios showcased the potential of the proposed formulation also in the more complex case of hybrid dynamics. Results show that the robustness reasoning added through SP makes it possible to find a combination of hardware, nominal trajectory and controller that reduces the cost of transport (CoT) in perturbed scenarios compared to an LQR-controlled monopod with decoupled co-design: the increase in CoT with respect to the nominal trajectory when performed without perturbations drops from 142% to 5.9%.

Despite the good results obtained with these three case studies, improving the scalability of this this co-design framework variant remains a challenge, mainly because the number of scenarios that can be assessed is limited by the feedback gains co-optimization. As stressed in [54], a potential solution may be the use of strategies derived from the Alternating Direction Method of Multipliers (ADMM) [97] and Bender's decomposition [98].

2.5 Conclusions

This chapter introduced a co-design framework for the energy efficiency analysis of a redundant actuation system. The analysis started with a basic case study that evaluated the energy consumption of an oscillating 1-DoF system. A frequency analysis was conducted to examine the energy consumption behavior of the redundant actuation system compared to a single SEA. Subsequently, a 2-DoF planar manipulator performing two tasks (swing-up maneuver and pick-and-place operation) was con-

sidered, and Co-Design was employed to optimize hardware and control choices. Finally, the closed-loop performance of the distinct actuation systems was assessed under modeling errors and disturbances using PD control to track the previously computed optimal state trajectories.

The results obtained from the study indicate that the proposed redundant actuation system is more energy-efficient than traditional actuators such as GMs or SEAs. When testing the system on a 1-DoF system performing a sinusoidal motion, energy savings compared to an SEA were significant, with savings sometimes exceeding 90%. For the 2-DoF system, the optimal configuration was found to consist of redundant actuation on the first joint and an SEA on the second. This configuration outperformed GMs, with energy consumption reductions of up to 99%, but led to limited savings (up to 7%) compared to using only SEAs. Task dependency was also observed in the energy savings of the 2-DoF system. Additionally, the study investigated the closed-loop behavior of the different actuation systems under modeling errors and disturbances using PD control to track optimal trajectories obtained through co-design. The results imply that the development of transmission systems, which relate the sinusoidal angle/torque input trajectory of the actuator to non-sinusoidal output trajectories, could be pursued to achieve substantial energy savings for other tasks. Moreover, the energy-efficient benefits of redundant actuation extend to closed-loop control with disturbances for a pick-and-place task. The proposed co-design framework enables not only the analysis and confirmation of the advantages in terms of energy efficiency of redundant actuation but also the identification of optimal values for the hardware parameters and state-control trajectories. Additionally, co-design analysis indicates that there is an increased incentive to use redundant actuation for joints closer to the base. Therefore, the importance of designing transmission systems to relay power over long distances will be even more critical for designs using redundant actuation schemes.

In order to enhance the analysis of closed-loop energy efficiency across various actuation systems, future investigations will explore the utilization of parametric optimization to calculate optimal feedback gains, as well as advanced control techniques such as MPC. Lastly, a mixed-integer optimal control problem could be formulated to enable the solver to determine which actuation system is optimal for each individual joint.

This chapter concludes by briefly introducing a method to extend such co-design framework to add robustness reasoning. This work was presented in a recent publication [54] first-authored by G. Bravo-Palacios

at the University of Notre Dame and co-authored by the author of this thesis. The proposed framework extension aims to enhance the robustness of robot motion planning by optimizing simultaneously the hardware, nominal trajectory, and feedback controller and by employing Stochastic Programming to produce robots that can handle external perturbations. The optimization problem is split into two stages, where the first stage optimizes variables that are invariant to uncertain events, such as hardware, nominal trajectory, and feedback gains. The second stage considers uncertain conditions caused by the stochastic application of disturbances and accounts for the feedback controller's action. Despite the two-stage abstraction, a single optimization problem is solved. Three case-studies were considered and all of them demonstrated the impact of hardware/morphology choices on performance, control actions, and energy efficiency. Moreover, the co-optimization of feedback gains turned out to play a crucial role in finding nominal trajectories that require less to be stabilized, thereby improving the robustness of the system. Although the three case-studies considered in [54] involve other robotic systems, such extension is straightforwardly applicable to the co-design framework reported in this chapter which was used to study the redundant actuation system. Currently, this robust variant of the co-design framework has two main limitations. First, the inclusion of more hardware parameters for optimization is limited by the trade-off between increased design space exploration and the resulting problem size and computation time. However, as certain combinations of hardware parameters may lead to poor quality local minima, this problem can be mitigated by requiring tighter bounds on such parameters to restrict the design space. Second, the co-optimization of feedback gains presented a challenge for problem scalability, limiting the number of scenarios that could be assessed through optimization. To address this challenge, a potential solution may be to consider decomposing and iteratively solving the co-design problem. In this direction, framework variants are being investigated that use approaches derived from the Alternating Direction Method of Multipliers [97] and Bender's decomposition [98].

Chapter 3

Towards global optimality

As demonstrated in chapters 2, when a model of the system to be controlled is available, one of the most powerful and flexible techniques to compute optimal trajectories is gradient-based Trajectory Optimization. Starting from a specific initial state, TO can find the control sequence that minimizes a cost function representing the task to be accomplished, where the system dynamics and possible state and control limits are considered as constraints. Such a powerful framework has led to excellent results when the problem is convex or slightly non-convex, especially when used in an MPC fashion. For example, it has been successfully employed to control high-dimensional non-linear systems such as quadrupeds and humanoids [99–102]. However, when the task to be accomplished requires a highly non-convex cost function and/or the dynamics is highly non-linear, the presence of multiple local minima, some of which are of poor quality (i.e., associated to a cost that is significantly worse than the global minimum), often prevents TO from finding a satisfying control trajectory. A possible solution to this problem is providing TO with a good initial guess, which turns out to be very complex—not to say impossible—without a deep prior knowledge about the system, which is often unavailable in practice. The IREPA algorithm [103] tackles this problem by building a kinodynamic Probabilistic Road Map (PRM) and approximating the *Value function* and control policy, which is then used to warm-start an MPC. The method produced satisfying results on a 3-DoF system, but it is limited to problems with a fixed terminal state, and

scaling to high dimensions seems not trivial due to its need to explicitly store the locally optimal trajectories in the PRM edges.

Approaches that can find the global optimum of non-convex problems exist, and are based on the Hamilton-Jacobi-Bellman equation [104] (for continuous-time problems) or Dynamic Programming (for discrete-time problems). However, these methods suffer from the curse of dimensionality, which restricts their applicability to systems with extremely few degrees of freedom. Some efficient solutions exist, but for specific problems, such as simple integrator dynamics with control-independent cost [105, 106]. Alternatively, the tensor-train decomposition [107] has been used to reduce the computational complexity of *value iteration*, by representing the *Value function* in a compressed form. However, the approach has been tested on stochastic optimal control problems with at most 7-dimensional state and scalar control. Subsequent improvements of that approach [108] could solve problems with a 12-dimensional state, but required some restrictions on the form of the dynamics and cost.

On the other hand, as mentioned in the previous chapters, in the recent years deep RL has shown impressive results on continuous state and control spaces, which represented its greatest challenge until 2015, when DDPG [5] was presented. Successively, many variants have been developed to further improve its performance, such as TD3, SAC, and RTD3 [17, 20, 109]. In 2021, Zhang *et al.* [110] managed to speed up DDPG's training time and enhance the effectiveness of its learning with their expansion called AE-DDPG, by making the agent latch on "good" trajectories very soon through the use of asynchronous episodic control and improving exploration with a new type of control noise. However, deep RL is intrinsically limited by its low sample efficiency, which implies the need for a considerable number of interactions with the environment to reach a good performance level.

To mitigate this problem, Levine *et al.* [111] proposed to guide the exploration process by using DDP as a generator of guiding samples that push the policy search towards low-cost regions. However, the *imitation* component of this approach makes its capability to find an optimal control policy strongly dependent on the quality of the guiding samples. This downside applies also to the method proposed by Mordatch and Todorov [112] combining policy learning and TO through the Alternating Direction Method of Multipliers (ADMM), which involves imitation in that the policy learning problem is reduced to a sequence of trajectory optimization and regression problems. In general, all the methods that are *imitation*-oriented (*e.g.* [113–115]) suffer from the same limitation:

the quality of what the RL algorithm can learn is limited by the quality of the demonstrations or guiding trajectories found by TO.

Recently, Morgan *et al.* [116] proposed an algorithm combining MPC and SAC that does not involve imitation and guarantees near-optimal performance if errors in the value function and model approximation are below a certain threshold. Even though this algorithm shares the same core idea of combining OC and RL as the one introduced in this chapter, they actually differ much in the way they learn a close-to-optimal policy. Indeed, the former merges MPC and whatever deep RL algorithm, which makes it very versatile but also inevitably inherit the problems of the selected RL algorithm as hyperparameters sensitivity or the need to learn also the Q-value function, whereas the algorithm presented here is stand-alone and requires the learning of only one critic representing the Value function.

In a similar spirit to [111], but with an imitation-free approach as [116], the algorithm presented in this section, named CACTO (Continuous Actor-Critic with TO), aims to mitigate both problems: the local minima issue affecting TO, and the low sample efficiency of RL. To do so, CACTO combines TO and RL in such a way that their interplay guides the search towards the globally optimal control policy.

3.1 CACTO: Continuous Actor Critic with TO

This section presents an optimization algorithm to solve a finite-horizon discrete-time OCP that takes the following general form:

$$\underset{X,U}{\text{minimize}} \quad J(X,U) = \sum_{k=0}^{T-1} l_k\left(x_k, u_k\right) + \; l_T\left(x_T\right) \tag{3.1a}$$

$$\text{subject to} \quad x_{k+1} = f_k(x_k, u_k) \quad \forall k = 0 \dots T-1 \tag{3.1b}$$

$$u_k \in \mathcal{U} \qquad \forall k = 0 \dots T-1 \tag{3.1c}$$

$$x_0 = x_{init} \tag{3.1d}$$

where the state and control sequences $X = x_{0\dots T}, U = u_{0\dots T-1}$, with $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$, are the decision variables. The cost function $J(\cdot)$ is defined as the sum of the running costs $l_k\left(x_k, u_k\right)$ and the terminal cost $l_T(x_T)$. The dynamics, control limits and initial conditions are represented by (3.1b), (3.1c) and (3.1d), with $\mathcal{U} = \{u \in \mathbb{R}^m : |u| \leq u_{max}\}$.

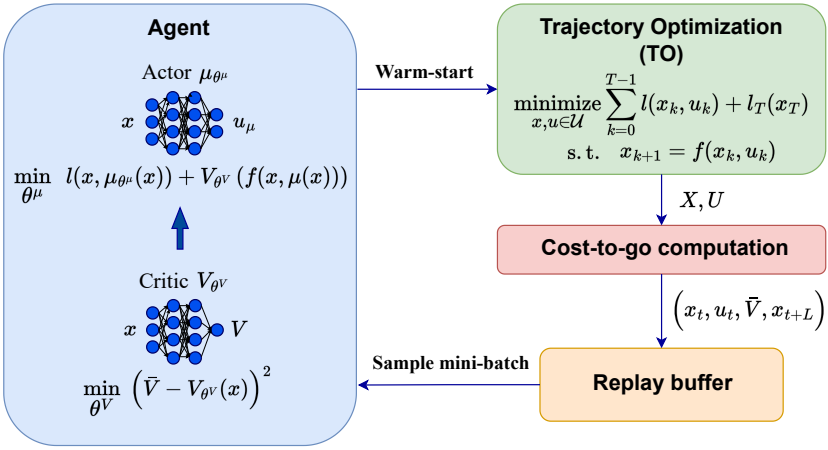


Figure 3.1: Scheme of CACTO: at each episode, a TO problem is warm-started with a policy rollout and solved, then the cost-to-go associated with each state of the trajectory found by TO is computed and the related transition is stored in the replay buffer. Finally, both critic and actor are updated by sampling a mini-batch of transitions from the replay buffer.

The algorithm presented in this section combines ideas from RL and TO. The core idea is to exploit TO to guide the exploration towards low-cost regions, so as to make the learning process more efficient. In turn, a rollout of the currently learned policy is used to initialize the next TO problem. In this way, the RL and TO components of the algorithm help each other, making convergence faster: as training proceeds, the TO solver is provided with better initial guesses and this increases the chance of obtaining better solutions; in turn, these trajectories drive the agent along lower-cost paths, pushing the critic towards a better approximation of the optimal *Value function* and, consequently, the actor towards the optimal policy. Fig. 3.1 shows a scheme of CACTO.

3.1.1 Algorithm description

As shown in Fig. 3.1, our algorithm can be broken down into four phases. In the TO phase (green block), the optimal state and control trajectories are computed considering a random initial state. Then, the cost-to-go is computed (red block) for each state of the optimal trajectory and stored

in a buffer (orange block). This buffer is then sampled to update both the critic and actor (blue block). Finally, to close the loop, a rollout of the actor is used to warm-start the next TO problem.

A more detailed description is reported in Algorithm 3. In the initial phase (lines 1-4), the critic $V(x|\theta^V)$ and actor $\mu(x|\theta^\mu)$ networks ("*Agent*" blue block in Fig. 3.1) are initialized, as well as the target critic network $V'(x|\theta^{V'})$ (copying the weights of the critic), and a buffer R is created. This buffer will store the transitions $(x_t, u_{TO,t}, \hat{V}_t, x_{\min(t+L+1,T)})$, where $\hat{V}_t$ is the partial cost-to-go until state $x_{\min(t+L+1,T)}$ after a rollout of either L steps, or the number of steps to reach T (whichever is lower). In this way, to update the critic one can choose between n-step Temporal Difference (TD) ($L = n-1$) and Monte-Carlo ($L = T-1-t$) by setting the hyperparameter L [117].

After the initialization phase, M episodes are performed (lines 5-29) starting from a random initial state x_0 (line 6). The state vector has a dimension of n and includes the time as its last component. The starting time index for each episode (and partial cost-to-go computation) is based on $x_0[n]$ (lines 14 and 17), so the length of each episode can vary. At the beginning of each episode, the state and control variables $(x^\circ_{TO}, u^\circ_{TO})$ of a TO problem are initialized (line 11) with a rollout $(x^\circ_\mu, u^\circ_\mu)$ of the policy network $\mu(x|\theta^\mu)$ (lines 7-10, "*Warm-start*" arrow in Fig. 3.1). The TO problem is then solved, the control inputs u_{TO} are applied starting from x_0, and the resulting costs are computed and saved (lines 14-16). Then, for each step of the episode the partial cost-to-go $\hat{V}_t$ is computed and the related transition is saved in R (lines 17-20, from "*Cost-to-go computation*" block to "*Replay Buffer*" one in Fig. 3.1). Finally, every e_{update} episodes the critic and the actor are updated: for K times a minibatch of S transitions is sampled from R (line 23, "*Sample mini-batch*" arrow in Fig. 3.1), for each of them the complete cost-to-go $\bar{V}_i$ is computed (line 24) by either adding the tail of the Value to $\hat{V}_i$ or copying the partial cost-to-go, which is the cost-to-go itself in case the lookahead window exceeds the episode length T. Then the critic and actor loss functions are minimized (lines 24-25). Considering that CACTO deals with finite-horizon problems, it is worth noting that V' is used only when $i + L + 1 \neq T$, namely when the costs-to-go are computed with n-step TD. The critic and actor loss functions are respectively the mean squared error between the costs-to-go and the values predicted by the critic $\left(\bar{V}_i - V(x_i)\right)^2$ and the Q-value, namely $Q(x_t, u_t) = l(x_t, u_t) + V(x_{t+1})$, which represents the policy's performance.

Algorithm 3: CACTO

Inputs: dynamics $f(\cdot,\cdot)$, running cost $l(\cdot,\cdot)$, terminal cost $l_T(\cdot)$, T, M, S, K, L
Output: trained control policy $\mu(x)$
$\theta^V \leftarrow$ random, $\theta^\mu \leftarrow$ random, $\theta^{V'} \leftarrow \theta^V$
Initialize replay buffer R
for *episode* $\leftarrow 1$ **to** M **do**
  $x_0 \leftarrow$ random, $x^\circ_{TO,0} \leftarrow x_0$
  for $t \leftarrow x_0[n]$ **to** T **do** *(policy rollout)*
    $u^\circ_{\mu,t} \leftarrow \mu(x^\circ_{\mu,t}|\theta^\mu)$
    $x^\circ_{\mu,t+1} \leftarrow$ Environment$(x^\circ_{\mu,t}, u^\circ_{\mu,t})$
  end
  $(x^\circ_{TO}, u^\circ_{TO}) \leftarrow (x^\circ_\mu, u^\circ_\mu)$ *(TO warm-start)*
  Solve TO problem and get control trajectory u_{TO}
  Agent's initial state $\leftarrow x_0$
  for $t \leftarrow x_0[n]$ **to** T **do** *(episode rollout)*
    $x_{t+1}, l_t \leftarrow$ Environment$(x_t, u_{TO,t})$
  end
  for $t \leftarrow x_0[n]$ **to** T **do**
    Compute partial cost-to-go: $\hat{V}_t = \sum_{j=t}^{\min(t+L,T-1)} l_j$
    $R \leftarrow (x_t, u_{TO,t}, \hat{V}_t, x_{\min(t+L+1,T)})$
  end
  if *episode* % $e_{update} = 0$ **then**
    for $k \leftarrow 1$ **to** K **do** *(critic & actor update)*
      Sample minibatch of S transitions $(x_i, u_{TO,i}, \hat{V}_i, x_{\min(i+L+1,T)})$
      Compute cost-to-go: $\bar{V}_i = \begin{cases} \hat{V}_i & \text{if } i+L+1 > T \\ \hat{V}_i + V'(x_{i+L+1}) & \text{otherwise} \end{cases}$
      Update critic by minimizing the loss over θ^V:
      $L_c = \frac{1}{S} \sum_{i=1}^{S} \left(\bar{V}_i - V(x_i|\theta^V)\right)^2$
      Update actor by minimizing the loss over θ^μ:
      $L_a = \frac{1}{S} \sum_{i=1}^{S} Q(x_i, \mu(x_i|\theta^\mu))$
      Update target critic: $\theta^{V'} \leftarrow \tau\theta^V + (1-\tau)\,\theta^{V'}$
    end
  end
end

3.1.2 Differences with respect to DDPG

This approach takes inspiration from DDPG, but presents a few key differences, which are highlighted in this subsection. A first difference is that the *Q-value* is replaced with the *Value function*, which is approximated by the critic network. By doing so, the complexity is reduced since the critic's input is only the state, thus there is no need to explore the control space. Keeping a *Q-value* approach instead, one could not explore the control space using only TO because only the locally-optimal control inputs would be chosen. Therefore, one should alternate the use of TO trajectories with other exploration techniques (*e.g.*, acting greedily *w.r.t.* the *Q-value function* and adding some noise), which would dilute the benefits of our algorithm, making it more similar to standard RL. Contrary to DDPG, this algorithm is on-policy, meaning that the critic estimates the Value of the exploratory policy being followed. This is the policy obtained by initializing TO with rollouts of the current policy network. This implies that it is important to size R not too large, so that only the most recent TO trajectories obtained by warm-starting TO with similar policies are stored; in this way, the critic, after its update, will approximate the *Value function* associated to that policy. Otherwise, the risk is having a critic that represents a meaningless *Value function*, because it used trajectories generated with very different policies.

Another difference with respect to DDPG is that CACTO considers finite-horizon problems because TO cannot be used to solve arbitrary infinite-horizon problems. Therefore, the *Value function* is time-dependent. This is addressed by considering time as the last component of the state vector ($x[n] = t$).

As in DDPG, also CACTO makes use of a target network V' to improve the stability of our algorithm. It is a copy of the critic network, whose weights $\theta^{V'}$ are updated slower than the critic's ones θ^V by performing only partially the update of the critic, that is $\theta^{V'} \leftarrow \tau\theta^V + (1-\tau)\,\theta^{V'}$ with $\tau \ll 1$ being the target learning rate.

Implementation Details

Each TO problem was solved using collocation, which was available in the *Pyomo* library and solved with the non-linear programming solver *IpOpt*.

The following details are not fixed as part of the proposed method. They are reported here for the sake of clarity and to help understand how the results in section 3.2 have been obtained. Concerning the critic's

neural network, a structure with two small preprocessing fully-connected layers with respectively 16 and 32 neurons is used, followed by two other fully-connected hidden layers with 256 neurons each, all with *ReLU* activation functions. The actor instead is represented by a residual neural network with two fully-connected hidden layers with 256 neurons each and *ReLU* activation functions, whose outputs are added and passed as inputs to the last layer with a *tanh* activation function to bound the final controls in $[-1, +1]$. A residual neural network is chosen to better propagate the gradient to the first layer and limit the effects of the potential combination of the vanishing gradient problem due to the *tanh* in the last layer and the dying ReLU problem [118] which would prevent the network from continuing to learn. Finally, the output of this last layer is multiplied by the control upper bound. To stabilize the training of these networks, the inputs are normalized and L2 weight and bias regularizers are used in each layer, with weight equal to 10^{-2}.

The critic and actor loss functions were minimized with a stochastic gradient descent optimizer *Adam* [119]. The maximum number of episodes is set to 80000 and stopped early the training of the neural networks if the results were satisfactory.

3.1.3 Global convergence proof for discrete spaces

As for most continuous-space RL algorithms, it is hard to give any formal guarantee of convergence for CACTO. However, we can show that, considering a discrete-space version of CACTO using look-up tables instead of DNN, the algorithm converges to a globally optimal policy. This version of CACTO performs sweeps of the entire state space as in classic DP algorithms (*e.g.*, Policy Iteration), rather than the asynchronous approach characterizing RL algorithms. Moreover, we consider the Policy Iteration version of our algorithm meaning that each phase of policy evaluation and policy improvement converges before the other begins. This proof does not extend easily to the original CACTO algorithm, but it gives us an insight into the soundness of its key principle.

Theorem 3.1. *Consider the following assumptions.*

- *State and control spaces are finite.*
- *The optimal Value function V^* is bounded.*
- *We have access to a discrete-space TO algorithm that can perform a local search and return a trajectory with a cost not greater than the cost of the initial guess.*

Let us define ${}^k\pi_{TO}$ *as the control policy obtained solving a TO problem using a rollout of the policy* ${}^k\pi$ *as initial guess. Then, starting from an arbitrary initial policy* ${}^0\pi$*, the following algorithm converges to the optimal Value function* V^* *and an optimal policy* π^**:*

$$
\begin{aligned}
{}^kV^{\pi_{TO}} &= \mathit{PolicyEvaluation}({}^k\pi_{TO}) \\
{}^{k+1}\pi &= \operatorname*{argmin}_{u\in\mathcal{U}}[l + {}^kV^{\pi_{TO}}]
\end{aligned}
$$

Proof. Let us define $\pi'(x)$ as the policy obtained by minimizing the *Action-value function* of π_{TO}, namely $Q^{\pi_{TO}}(x,u)$:

$$
\pi'_t(x_t) \triangleq \operatorname*{argmin}_{u\in\mathcal{U}}[l_t(x_t,u) + V^{\pi_{TO}}_{t+1}(x_{t+1})] \quad \forall\, x_t, t \tag{3.2}
$$

where $x_{t+1} = f_t(x_t,u)$, and $V^{\pi_{TO}}_{t+1}(x_{t+1})$ is the cost-to-go following π_{TO} from x_{t+1} at time $t+1$.

Since TO always finds a solution that is at least as good as the provided initial guess, we have that:

$$
V^{\pi}_t(x) \geq V^{\pi_{TO}}_t(x) \quad \forall\, x, t \tag{3.3}
$$

Since π' is the minimizer of $Q^{\pi_{TO}}$ (see (3.2)), we know that:

$$
V^{\pi_{TO}}_t(x) = Q^{\pi_{TO}}_t(x, \pi_{TO,t}(x)) \geq Q^{\pi_{TO}}_t(x, \pi'_t(x)) \tag{3.4}
$$

Starting from (3.4), and following the same idea of the convergence proof of Policy Improvement [120], we can write:

$$
\begin{aligned}
V^{\pi_{TO}}_t(x_t) &\geq \min_{u\in\mathcal{U}}[l_t(x_t,u) + V^{\pi_{TO}}_{t+1}(x_{t+1})] && (3.5)\\
\text{(by (3.2))} \;&=\; l_t(x_t,\pi'_t(x_t)) + V^{\pi_{TO}}_{t+1}(x_{t+1}) && (3.6)\\
\text{(by (3.4))} &\geq l_t(x_t,\pi'_t(x_t)) + Q^{\pi_{TO}}_{t+1}(x_{t+1},\pi'_{t+1}(x_{t+1})) && (3.7)\\
&= l_t(x_t,\pi'_t(x_t)) + [l_{t+1}(x_{t+1},\pi'_{t+1}(x_{t+1})) && (3.8)\\
&\qquad + V^{\pi_{TO}}_{t+2}(x_{t+2})] \\
&\;\;\dots \\
&\geq \sum_{k=t}^{T-1} l_k(x_k,\pi'_k(x_k)) + l_T(x_T) \triangleq V^{\pi'}_t(x_t) && (3.9)\\
\text{(by (3.3))} &\geq V^{\pi'_{TO}}_t(x_t) && (3.10)
\end{aligned}
$$

To infer (3.9) from (3.8) we can iteratively apply the same reasoning we used to go from (3.6) to (3.8). This proves that the updated policy

Table 3.1: Time, number of DNN updates, number of environment steps, and learning rates (LR_C for critic and LR_A for actor) used for each test.

System	***Time [h]***	***# updates***	***# env. steps***	$\boldsymbol{LR_C}$, $\boldsymbol{LR_A}$
Single Int.	5.46	$110k$	$3.4M$	$5e^{-3}$, $1e^{-4}$
Double Int.	7.29	$130k$	$4.1M$	$5e^{-3}$, $5e^{-4}$
Dubins Car	10.45	$260k$	$4.1M$	$1e^{-3}$, $5e^{-4}$
Manipulator	30.68	$385k$	$6M$	$1e^{-3}$, $5e^{-5}$

π'_{TO} cannot be worse than the previous one π_{TO}. Since $V^{\pi_{TO}}$ is non-increasing and always bounded from below by the optimal value V^*, it follows from the *monotone convergence theorem* that $V^{\pi_{TO}}$ converges to a constant value V^∞. At convergence we must have:

$$V_t^\infty(x_t) = \min_{u \in \mathcal{U}} [l_t(x_t, u) + V_{t+1}^\infty(x_{t+1})] \quad \forall\, x_t, t$$

This is Bellman's optimality equation, which is a sufficient condition for global optimality, so it follows that the algorithm converges to the optimal Value ($V^\infty = V^*$) and to an optimal policy ($\pi^\infty = \pi^*$). □

3.2 Results

This section presents the results of four different systems of increasing complexity: single integrator, double integrator, Dubins car, and 3-joint planar manipulator. For each system, the task consists of finding the shortest path to a target point (related to the end-effector's position for the manipulator) while ensuring low control effort and avoiding an obstacle. The aim is verifying the capability of CACTO to learn a control policy to warm-start TO so that it can find "good" trajectories, where other warm-starting techniques, such as using the initial conditions (ICS) or random values, would make it converge to poor local minima. More precisely, the ICS warm-start uses the initial state (varying at each episode) as initial guess for the state variables in the OCP, for all time steps, and 0 as initial guess for the control variables.

For each system, the XY-plane is divided in a grid of 961 points and, starting from each point with 0 initial velocity, the results of TO when warm-started with either the policy learned by CACTO, or a random initial guess, or the ICS for x and y (and 0 for the remaining variables)

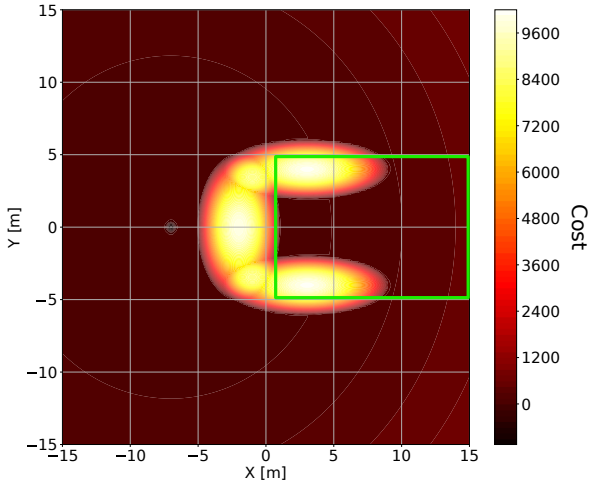


Figure 3.2: Cost function without the control effort term (3.15), considering a target point located at $[-7, 0]$ with weights $w_d = 100$, $w_p = 5 \cdot 10^5$ and $w_{ob} = 1 \cdot 10^6$. The green rectangle delimits the *Hard Region*.

are compared. Table 3.1 reports the time and number of updates needed to train the critic and the actor for each test.

In the four tests, the dynamics of the system under analysis is changed keeping the same highly non-convex cost function to ensure the presence of many local minima, that takes the following form:

$$l(\cdot) = \frac{1}{c_2}\left(\sum_{i=1}^{4} l_i(\cdot) - c_1\right) \tag{3.11}$$

$$l_1(\cdot) = w_d((x - x_g)^2 + (y - y_g)^2) \tag{3.12}$$

$$l_2(\cdot) = \frac{w_p}{\alpha_1}\ln(e^{-\alpha_1\left(\sqrt{(x-x_g)^2+c_2}+\sqrt{(y-y_g)^2+c_3}+c_4\right)} + 1) \tag{3.13}$$

$$l_3(\cdot) = \frac{w_{ob}}{\alpha_2}\sum_{i=1}^{3} -\ln\left(e^{-\alpha_2\left(\frac{(x-x_{ob,i})^2}{(a_i/2)^2}+\frac{(y-y_{ob,i})^2}{(b_i/2)^2}-1\right)} + 1\right) \tag{3.14}$$

$$l_4(\cdot) = w_u||u||_2^2 \tag{3.15}$$

where (x_g, y_g) are the target point coordinates, $c_1 = 10000$ and $c_2 = 100$ are two constants, while $c_3 = 0.1$, $c_4 = -2c_3 - 2\sqrt{c_3}$, $\alpha_1 = 50$ and $\alpha_2 = 50$ are the parameters that define the smoothness of the *softmax* functions used to model respectively a cost valley in the neighborhood of the target

and the three ellipses (centered at $(x_{ob,i}, y_{ob,i})$ and with principal axes a_i and b_i) forming the obstacle. The four terms composing the cost describe the task to be performed: (3.12) and (3.13) push the agent to reach the target point (with weights w_d and w_p, respectively), (3.14) makes it avoid the C-shaped obstacle represented by three overlapping ellipses (with weight w_{ob}) and (3.15) discourages it from using too much control effort (with weight w_u). Fig. 3.2 illustrates the cost function without the control effort term (3.15), where the cost peaks correspond to the obstacle penalties (3.14) and the cost valley to the term (3.13). Table 3.1 reports the time and number of updates needed to train the critic and the actor for each test.

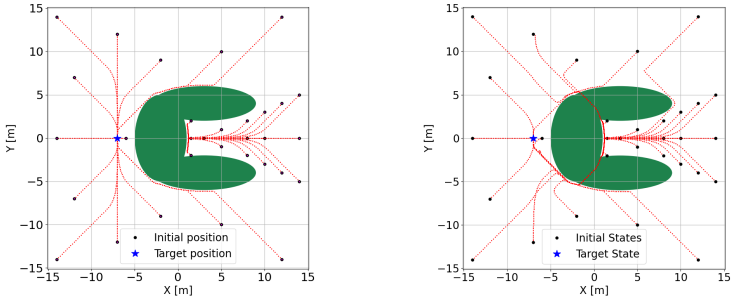
The critic is updated with Monte-Carlo (MC) for the 2D point, and with 50-step TD for the car and the manipulator. The reason for using TD rather than MC lies in the stability of the training of critic and actor. Indeed, in the early phase of training, using Monte-Carlo could lead to large variations of the critic, and indirectly also of the actor, because the target Values would be extremely different from the current Values. Moreover, in the early training phase TO could compute extremely poor trajectories because of the poor initial guess provided by the actor. In turn, these poor trajectories result in hard-to-learn *Value functions*. Therefore, it is empirically observed that these two effects can destabilize the training, leading to either longer training times, or even divergence of the algorithm.

To summarize the results, Table 3.2 reports the number of times that CACTO made TO find lower-cost solutions than the other two warm-starts, considering both the whole grid and the region from which it is harder for TO to find "good" solutions (*Hard Region*).

3.2.1 Single Integrator

The first problem that is considered is a simple 2D single integrator that has to reach a target point avoiding a C-shaped obstacle. The state is $[x, y, t] \in \mathbb{R}^3$, while the controls are the 2D velocities $[v_x, v_y] \in \mathbb{R}^2$, bounded in $[-4, 4]\frac{m}{s}$.

The task is simple except when the system starts from the *Hard Region* ($x \in [1, 15]$ m and $y \in [-5, 5]$ m), where TO can easily get stuck in local minima. Most of the times this means that the resulting trajectories point immediately towards the target, making the 2D point stay at the right boundary of the vertical ellipse, as in the case with ICS warm-start illustrated in Fig. 3.3(a). But since the C-shaped obstacle is modeled



(a) ICS warm-start (b) Random warm-start

Figure 3.3: Optimal trajectories of the 2D single integrator obtained with ICS and random warm-starts.

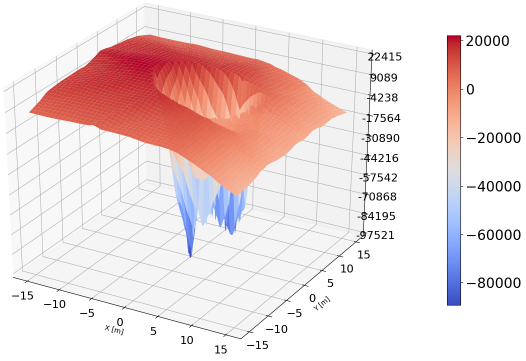


Figure 3.4: Single integrator: 3D-colormap of the Value function approximated by the critic after 110k updates considering t=0s.

as three overlapping ellipses represented by three soft penalties (3.14) in the cost function, it can also occur that the 2D point passes through them to reach the target, albeit at high cost, as shown in Fig. 3.3(b). Table. 3.2 reports the percentage of the time that warm-starting TO with CACTO rollouts leads to lower costs than using random values and the initial conditions for x and y and 0 for the remaining variables as initial guess. CACTO warm-start wins over the other two techniques, particularly if considering the agent starting from the *Hard Region* where warm-starting TO with CACTO leads to lower-cost solutions 99.11%

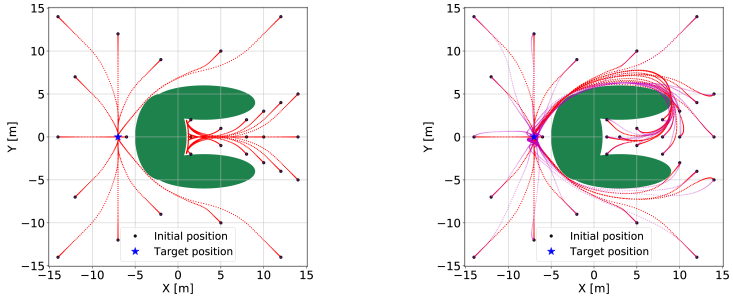
Table 3.2: Percentage of the time that warm-starting TO with CACTO leads to lower costs than using random initial guesses (CACTO vs. Random) and the initial conditions for x and y and 0 for the remaining variables (CACTO vs. ICS) as initial guess. Also the percentages when CACTO has lower or equal cost (*i.e.*, including ties) as its competitor are reported. The best result out of 5 runs is reported for random warm-start. For each system, a 31x31 grid is sampled for the initial x and y coordinates (those of the end effector for the manipulator) and setting the remaining initial state components to 0 (the joint positions of the manipulator are obtained by fixing the orientation of the end-effector and inverting the kinematics). The *Hard Region* is the region delimited by $x \in [1, 15]$ m ($[1, 23]$ m in the manipulator test) and $y \in [-5, 5]$ m.

System	**Whole Space**		**Hard Region**	
	$<$ ($\leq$) ***Random***	$<$ ($\leq$) ***ICS***	$<$ ($\leq$) ***Random***	$<$ ($\leq$) ***ICS***
Single Int.	99.88% (99.88%)	14.49% (99.88%)	99.11% (99.11%)	91.96% (99.11%)
Double Int.	99.88% (99.88%)	12.38% (99.88%)	99.11% (99.11%)	91.96% (99.11%)
Dubins Car	89.72% (98.83%)	15.65% (95.56%)	100% (100%)	92.86% (100%)
Manipulator	91.78% (91.91%)	77.94% (78.33%)	87.50% (87.50%)	100% (100%)

and 91.96% of the time compared to using random values and the ICS as the initial guess, respectively. Indeed, as it can be noticed in Fig. 3.4, at the end of the training the critic gives a good approximation of the globally optimal *Value function*, whose gradient in the *Hard Region* does not point towards the right boundary of the vertical ellipse but to the opposite direction. As the actor's updates follow the critic's gradient during training, this allows the actor to learn how to make the agent escape that region.

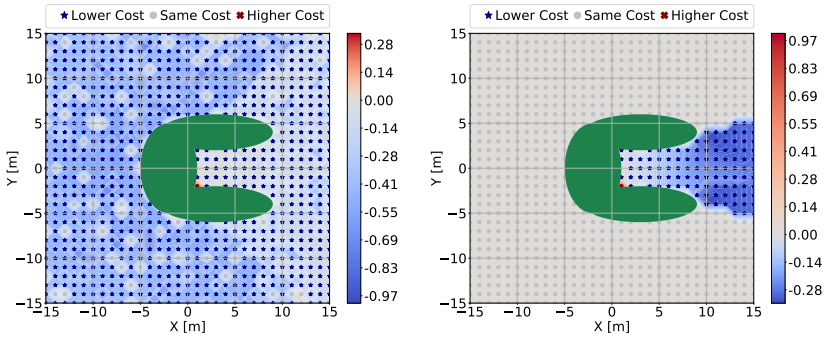
3.2.2 Double Integrator

The same experiment is run with the 2D point considering a double integrator dynamics. Therefore, now the state includes also the veloc-



(a) ICS warm-start (b) CACTO warm-start

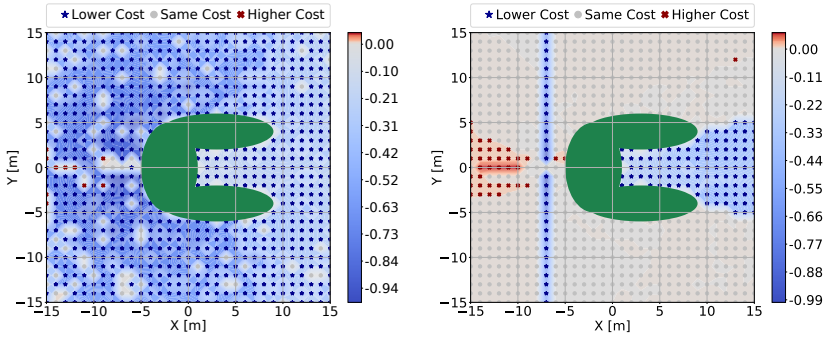
Figure 3.5: Optimal trajectories (red) of the 2D double integrator obtained with ICS and CACTO warm-starts. In (b), the magenta lines represent the CACTO policy rollouts.



(a) CACTO vs. Random warm-starts (b) CACTO vs. ICS warm-starts

Figure 3.6: Double integrator: cost difference between CACTO warm-start and other two warm-starts normalized by the largest cost difference.

ities $[x, y, v_x, v_y, t] \in \mathbb{R}^5$ and the control inputs are the accelerations $[a_x, a_y] \in \mathbb{R}^2$. As illustrated in Fig. 3.5(b), the rollouts of the CACTO policy are already close to the globally optimal trajectories, therefore TO only needs to refine them when they are used as initial guess. Trajectories are referred to as globally optimal if their cost is the lowest among those obtained solving several TO problems warm-started with random initial guesses. Fig. 3.6 shows instead the cost difference (normalized



(a) CACTO vs. Random warm-starts

(b) CACTO vs. ICS warm-starts

Figure 3.7: Dubins car model: cost difference between CACTO warm-start and other two warm-starts normalized by the largest cost difference.

by the highest difference in absolute value) when TO is warm-started with rollouts of the CACTO policy in place of random values or the ICS, respectively. Fig. 3.6(a) clearly shows that using rollouts of the policy learned by CACTO as an initial guess makes TO find lower-cost solutions from almost any initial state compared to those found with a random initial guess. In Fig. 3.6(b) instead, it can be noticed that ICS are a good initial guess for the majority of initial states and TO finds the same solutions as when warm-started with CACTO rollouts. However, when starting the agent in the *Hard Region*, the ICS warm-start makes TO find poor local minima, where the agent remains stuck in that region or passes through the obstacle, whereas CACTO enables TO to successfully bring the agent to the target without touching the obstacle. Also in this case, in the *Hard Region*, warm-starting TO with CACTO rollouts rather than with random values or ICS makes TO find lower costs with the same percentage as in the previous test.

3.2.3 Dubins Car

To test CACTO with a higher-dimensional system, a jerk-controlled version of the so-called *Dubins car model* [121] is selected, while keeping the same environment and cost function of the previous tests. Now the state has size 6 because it includes the steering angle θ, the tangential velocity v, and acceleration a, in addition to the coordinates x and y of the

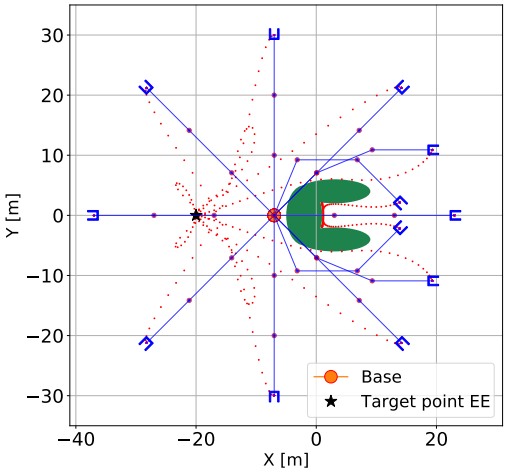


Figure 3.8: TO solutions considering 12 different initial configurations of the manipulator and ICS warm-start. The red dotted lines represent the trajectories performed by the end-effector (EE).

car's center of mass and time t: $s = (x, y, \theta, v, a, t) \in \mathbb{R}^6$. The control is still bi-dimensional and it consists in the steering velocity and the jerk $[\omega, j] \in \mathbb{R}^2$. When the car starts from the *Hard Region*, TO warm-started with CACTO always finds a lower cost than that obtained by warm-starting TO with random values, and it does so the 92.86% of the time when compared to using ICS warm-start, as reported in Table 3.2. In addition, Fig. 3.7(b) shows that TO warm-started with ICS is not able to find the globally optimal solution also when the car starts from a point along the vertical line passing through the target point. This is due to the fact that in that region the gradient of the cost is zero along the initialization itself.

3.2.4 3-DoF Planar Manipulator

Finally, CACTO is tested on a problem with a 7D state and 3D control space. It consists of a 3-DoF planar manipulator with base fixed at $[-7, 0]$ m, working in the same environment of the previous tests, whose end-effector has to reach a target point located at $x_g = -20$ m and $y_g = 0$ m. The cost function is always (3.11), where x and y represent the coordinates of the end-effector. Fig. 3.8 shows some solutions found

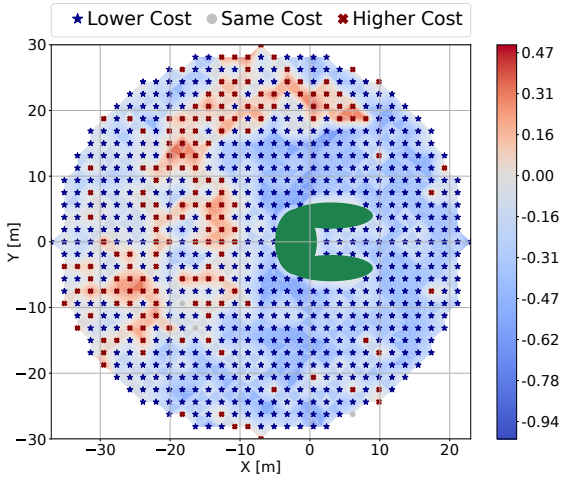


Figure 3.9: 3-DoF Manipulator: normalized cost difference when warm-starting TO with CACTO compared to using the initial conditions as initial guess for the joint positions and 0 for the remaining variables.

by TO when warm-started with the ICS. It may seem that TO succeeds in some cases in finding the globally optimal trajectories for those initial configurations, but actually all of them are only locally optimal, as shown by the negative cost difference in Fig. 3.9 when those solutions are compared to the ones obtained by warm-starting TO with CACTO. This test is harder, meaning that it is much easier for TO to find poor local minima, not only due to the larger state-action space, but also because the actor has to intrinsically learn the manipulator kinematics. Indeed, considering the whole manipulator workspace, CACTO warm-start wins over using the ICS or random values 77.94% and 91.78% of the time, respectively, as reported in Table 3.2.

3.2.5 Comparison with DDPG and PPO performance

To compare CACTO and DDPG, a custom implementation of DDPG with hand-tuned hyper-parameters (DDPG-c) is used, as well as that from Stable Baselines (DDPG-sb), to find a warm-starting policy for the 2D double integrator test of section 3.2.2. In addition, the comparison is extended also to PPO from Stable Baselines. Fig. 3.10 shows the costs

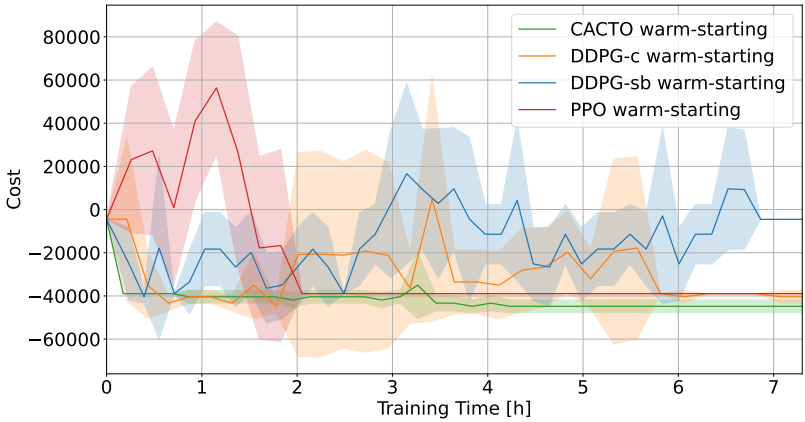


Figure 3.10: Cost of the trajectory found by TO (starting the 2D point from $[5, 0]$m with 0 velocity) when warm-started with rollouts of the policy learned with CACTO (green), custom DDPG (DDPG-c in orange), Stable Baselines' DDPG (DDPG-sb in blue) and PPO (red) as their trainings proceed. The shaded area denotes the standard deviation over 5 different runs.

obtained by TO as functions of the computation time allocated to each algorithm to train its warm-starting policy. Clearly, CACTO was faster than the other RL algorithms in learning a policy enabling TO to find lower-cost solutions, and its training was also more stable (rollouts with lower variance).

Besides this quantitative comparison, one could also try to qualitatively compare the results reported in the previous sections with the ones reported for DDPG in [5]. Among their tests, the most similar to the ones here is the *fixedReacher*, where a 3-DoF arm must reach a fixed target; this has the same dimensions as the manipulator test. However, the cost/reward used here function is highly non-convex, particularly due to the obstacle avoidance term, while theirs was quadratic and consisted of only two terms. This makes it harder to reach convergence in the DNN training. Using CACTO in such a simple setting, with a convex cost-function, would not make sense: TO would converge to the global optimum even with a trivial warm-start (indeed DDPG converged to roughly the same policy found by iLQG). Consequently, it makes no sense to use these results for a comparison with CACTO, which should

instead be based on highly non-convex problems, where TO cannot find the global optimum with a naive initial guess.

3.3 Potential improvements

Despite the encouraging results, CACTO can still be improved, in particular concerning its computation time. This can be achieved by some different techniques. For instance, a possible direction to speed up the learning of the critic is to use Sobolev Training [122], so as to exploit the derivatives of the *Value function* computed by a DDP-like TO algorithm [55]. With Sobolev Training, the critic is updated by minimizing the weighted sum of the mean squared error between the predicted and computed cost-to-go and some norm of their gradients. The critic's update is thus more informative, which means that one could afford to choose a higher learning rate or a smaller minibatch size. In turn, as the critic reaches faster a good approximation of the *Value function*, the actor will also be faster in learning a good control policy to warm-start TO. Considering that CACTO is based on the interplay between TO and RL, this should make it converge faster to the globally optimal policy. However, there may be potential drawbacks related to this approach, that are the addition of a new hyperparameter that may not be easy to tune, since it would depend on the problem considered, and stability issues that typically occur when updating deep neural networks too aggressively.

One could also consider to implement CACTO using sampling-based multi-query planners, similarly in spirit to what [103] proposes with PRMs or [123] with RRTs. This could also accelerate the learning process, but to the detriment of the applicability of the algorithm beacuse of two reasons. First, such sampling-based planners need a target state to reach, whereas CACTO works simply with a cost function. Even though in the tests reported in this chapter the cost function always encodes a target state, this is not at all a requirement for CACTO. Second, CACTO produces a continuous policy, which can be used from any state, whereas sampling-based algorithms produce a graph with a limited number of nodes. This means that if the agent starts from a state that is not part of the graph, it first needs to connect the current state with one of the nodes of the graph, resulting in extra computations for online applications.

In addition, also using a prioritized replay buffer (*e.g.*, w.r.t. the

TD error) or the Hindsight Replay Buffer (HER) [124] may be beneficial for speeding up the learning process. From preliminary tests, it was observed that the use of a prioritized replay buffer strongly impacts the training stability, as the state space is no longer sampled uniformly and so the value in some regions may diverge. Therefore, other additional techniques may be needed to compensate such instability which deserve further investigation.

Another potential improvement could be the use of randomized smoothing [125] to manage non-differentiability in the dynamics of the system when it becomes hybrid due to contacts, as it occurs in legged locomotion or manipulation problems for instance.

Regarding the practical use of CACTO, besides using it as initial guess provider for TO, where the RL and TO environments must match, it would be interesting to use it as a deep RL technique to find directly a control policy, where the two environments do not need to match (*e.g.*, the TO problem could be a simplified version of the RL environment without noise sources).

Finally, CACTO will be extended to include also the optimization of hardware parameters, so as to create a co-design framework which is robust against the local minima problem.

Chapter 4

Conclusions

This thesis dealt with the concurrent design (Co-Design) of hardware and control for robots and it can be divided into two parts. The first part highlighted the importance and effectiveness of following a co-design approach when benchmarking different robotic systems, specifically when carrying out a comparison in terms of energy efficiency between different actuation systems. The second part tackled the issue of convergence to "poor" local minima, which typically occurs with highly non-convex optimal control problems as those of Co-Design, by proposing an algorithm to learn a "good" control policy to be used as initial guess provider for trajectory optimization problems. Even though the current stage of this algorithm considers only control in the optimization process, the next step will be to extend it by including also the hardware optimization so as to enable its applicability to co-design problems.

4.1 Summary

Chapter 2 presented an application of Co-Design for the energy efficiency analysis of a redundant actuation system consisting in a series elastic actuator (SEA) working in parallel with a quasi-direct drive (QDD) to actuate the same joint. This kind of actuator was firstly introduced in the late 90', but it was not clear yet whether it was energetically more convenient than a SEA or a geared motor (GM) working alone. The analysis started from a simple case-study, which examines the energy consumption of an oscillating 1-DoF system. Then, the analysis focused on a 2-DoF planar manipulator. Two tasks were considered and Co-

Design was used to make optimal choices in the simultaneous design of hardware and control. Finally, the closed-loop behavior of the different actuation systems subject to modeling errors and disturbances was investigated using PD control to track the previously computed optimal trajectories. The results showed that the proposed redundant actuation is energetically convenient compared to standard actuators as GMs or SEAs. With the 1-DoF system performing a sinusoidal motion, the energy savings compared to an SEA can be very large, sometimes exceeding 90%. For the 2-DoF system instead, the energy savings turned out to be task dependent, but in any case the optimal configuration seems to consist of having redundant actuation on the first joint and an SEA on the second one. This system can outperform GMs (up to 99% energy consumption reduction), but it leads to limited savings (up to 7%) with respect to only SEAs. This finding suggests that transmission designs (e.g, as in [126]) relating sinusoidal angle/torque input trajectory of the actuator to nonsinusoidal output trajectories might be pursued to enable significant energy savings for other behaviors. In addition, for a pick-and-place task, the energetic advantages of the redundant actuation also extend to closed-loop control with perturbations. With the proposed co-design framework, it was possible not only to study and verify the energy convenience of this redundant actuation, but also to obtain optimal values for its hardware parameters and optimal state-control trajectories. This gives flexibility in the design process, since with little changes to the framework (*e.g.*, to constraints, objective function etc.) one can obtain new designs of a robotic system tailored to different specific tasks. Furthermore, Co-Design showed that there is increased incentive for adopting redundant actuation for more proximally located joints. This suggests that the importance of designing transmission systems to relay power distally will be even more important for designs adopting redundant actuation schemes.

Chapter 3 presented a new algorithm (CACTO) for finding quasi-optimal control policies, whose rollouts are meant to be used as initial guess for trajectory optimization problems. In particular, this work addressed the open problems affecting TO and Deep Reinforcement Learning (RL): the possibility of getting stuck in poor local minima when TO is not properly warm-started on one side, and the low sample efficiency (in addition to the strong dependence on the exploration process) of Deep RL. The proposed algorithm relies on the combination of TO and Deep RL in such a way that their interplay guides in an efficient way the RL exploration process towards a globally optimal control policy.

A proof of policy improvement for a discrete-space version of the algorithm was provided, which gives insight into its underlying theoretical principles. The effectiveness of the algorithm was shown by testing it on four systems of increasing complexity, with highly non-convex cost functions, where TO struggles to find "good" solutions. More precisely, the systems considered are a 2D point with single and double integrator dynamics, a Dubins car and a 3-link manipulator, and the task to be accomplished consists of finding the shortest path to a target point while ensuring low control effort and avoiding an obstacle. Using rollouts of the policy learned by CACTO to warm-start the TO problem of interest outperformed standard warm-starting techniques in all tests. Moreover, it was demonstrated that CACTO is faster than other state-of-the-art RL algorithms in learning a policy enabling TO to find lower-cost solutions, and that its training is also more stable (rollouts with lower variance). Overall, even though preliminary, the results validated the proposed methodology and unlocked a wide range of possible applications.

4.2 Discussion and future work

This section points out the potential impact of this thesis for the scientific community, stressing the limitations and the possible future developments of the presented methods and algorithms. The work presented in this thesis advanced the current state of the art in the field of Co-Design. Hopefully, the efforts put in this work represent another step towards the widespread adoption of Co-Design for the design of hardware and control of high-performance robotic systems.

Concerning the co-design application introduced in chapter 2, the main implication of the reported results is that such redundant actuation system should be further analyzed considering other systems and tasks where energy efficiency is of fundamental importance, for instance in rescue or delivery applications of battery-powered legged robots or energy harvesting applications with other devices than wave energy generators [92, 93], particularly when the kinetic energy source generates sinusoidal movements. More generally, given the flexibility and capability of such framework to compute optimal hardware-control pairs in those scenarios where human intuition does not suffice, it can be used to fairly benchmark novel actuation systems against already existing ones by changing only some constraints of the OCP.

However, the main limitation of this work is represented by the potential presence of many different "poor" local minima, which may prevent the solver from finding the most performing pair of hardware and control strategy for each actuation system, thus making the comparison misleading. A solution to this issue could be the application of CACTO (chapter 3) to such optimal control problem as soon as it will be extended to include also the hardware optimization. Moreover, to improve the analysis of closed-loop energy efficiency for different actuation systems, future work could consider also the optimization of the feedback gains, as done in [54] and briefly described at the end of chapter 2, as well as the use of more advanced control methods such as Model Predictive Control. Another potential future development is the extension of this study to a single leg of a quadruped subject to different gaits, obtaining optimal hardware designs and control trajectories via simulation, and analyzing the energy efficiency of diverse actuation configurations in a real application. Finally, a mixed-integer OCP could be formulated to let the solver choose which actuation system is the best to actuate each single joint.

The algorithm presented in chapter 3 (CACTO) can have important consequences in the robotics field and significant impact on future works. Considering its current development stage, it can be used without much modifications to find a "good" warm-starting strategy for any OCP involving very non-convex objective functions, which may represent multi-goal tasks, and systems with state-action space that is not of high-dimensionality. This use of CACTO can help researchers increase the confidence level about the global optimality of the solutions of their trajectory optimization problems, which is currently not achievable with any other tool or technique. This is not limited to the robotics community as CACTO is application-agnostic, indeed it can be applied in any field where it is required to optimize state and control trajectories generated by whatever system whose dynamics is differentiable. Nonetheless, its greatest implication is that it paves the way for the design of a family of algorithms merging model-based and model-free techniques, which is a research direction that is gaining increasingly traction among roboticists in the very recent years.

Despite the encouraging results, CACTO can still be improved, in particular concerning its computation time. A potential solution that is currently being investigated to speed up the learning of the critic is using Sobolev Training [122]. This technique pushes the critic to match also the gradient of the *Value function*, which can be easily computed

by any DDP-like TO algorithm, such as [55]. Moreover, besides using CACTO as initial guess provider for TO, where the RL and TO environments must match, it is worth testing it also as a deep RL technique to find directly a control policy, where the two environments do not need to match (*e.g.*, the TO problem could be a simplified version of the RL environment without noise sources). Finally, the most promising development direction for CACTO is its extension to co-design problem by including also the hardware optimization, so as to propose a co-design framework robust to local minima, which is a crucial issue in co-design applications.

Bibliography

[1] G. Grandesso, G. Bravo-Palacios, P.M. Wensing, M. Fontana, and A. Del Prete. Exploring the limits of a redundant actuation system through co-design. *IEEE Access*, 9:56802–56811, 2021.

[2] © 2023 IEEE. Reprinted, with permission, from G. Grandesso, E. Alboni, G.P. Rosati Papini, P. Wensing, and A. Del Prete. CACTO: Continuous Actor-Critic with Trajectory Optimization—Towards global optimality. *IEEE Robotics and Automation Letters*, 8(6):3318–3325, 2023.

[3] M. Bjelonic, C.D. Bellicoso, Y. de Viragh, D. Sako, F.D. Tresoldi, F. Jenelten, and M. Hutter. Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots. *IEEE Robotics and Automation Letters*, 4(2):2116–2123, 2019.

[4] P.I. Corke. A simple and systematic approach to assigning denavit–hartenberg parameters. *IEEE Transactions on Robotics*, 23(3):590–594, 2007.

[5] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2015. [Online]. Available: *arXiv:1509.02971* [cs.LG].

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*, pages 53–54. The MIT Press, second edition, 2018.

[7] D.P. Bertsekas. *Reinforcement Learning and Optimal Control*, pages 7–12. Athena Scientific, first edition, 2019.

[8] D.P. Bertsekas. *Reinforcement Learning and Optimal Control*, pages 200–208. Athena Scientific, first edition, 2019.

[9] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*, page 74. The MIT Press, second edition, 2018.

[10] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*, page 93. The MIT Press, second edition, 2018.

[11] D. Silver, G. Lever, N. Heess, T. Degris, D. Wiestra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML '14: Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 387–395, Beijing, China, 2014.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning, 2013. [Online]. Available: *arXiv:1312.5602* [cs.LG].

[13] C.J.C.H. Watkins and P. Dayan. Design for control-a concurrent engineering approach for mechatronic systems design. *Machine Learning*, 8:279–292, 1992.

[14] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*, pages 264–265. The MIT Press, second edition, 2018.

[15] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration, 2018. [Online]. Available: *arXiv:1812.02900* [cs.LG].

[16] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay, 2015. [Online]. Available: *arXiv:1511.05952* [cs.LG].

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. [Online]. Available: *arXiv:1801.01290* [cs.LG].

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. [Online]. Available: *arXiv:1707.06347* [cs.LG].

[19] J. Schulman, S. Levine, P. Moritz, M.I. Jordan, and P. Abbeel. Trust region policy optimization, 2015. [Online]. Available: *arXiv:1502.05477* [cs.LG].

[20] H. Van Hoof S. Fujimoto and D. Meger. Addressing function approximation error in actor-critic methods, 2018. [Online]. Available: *arXiv:1502.05477* [cs.LG].

[21] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. [Online]. Available: *arXiv:1602.01783* [cs.LG].

[22] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Stable baselines, 2023. [Online]. Available: `https://github.com/DLR-RM/stable-baselines3`.

[23] K. Sims. Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, Orlando, Florida, 1994.

[24] Q. Li, W. J. Zhang, and L. Chen. Design for control-a concurrent engineering approach for mechatronic systems design. *IEEE/ASME Transactions on Mechatronics*, 6(2):161–169, 2001.

[25] J. Hass, J. M. Herrmann, and T. Geisel. Optimal mass distribution for passivity-based bipedal robots. *The International Journal of Robotics Research*, 25(11):1087–1098, 2006.

[26] K. Mombaur. Using optimization to create self-stable human-like running. *Robotica*, 27(3):321–330, 2009.

[27] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *GECCO '13: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 167–174, Amsterdam, The Netherlands, 2013.

[28] K.M. Digumarti, C. Gehring, S. Coros, J. Hwangbo, and R. Siegwart. Concurrent optimization of mechanical design and locomotion control of a legged robot. In *CLAWAR '14: 17th International*

Conference on Climbing and Walking Robots, pages 315–323, Poznan, Poland, 2014.

[29] J. T. Allison. Engineering system co-design with limited plant redesign. *Engineering Optimization*, 46(2):200–217, 2014.

[30] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane. Task-based limb optimization for legged robots. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2062–2068, 2016.

[31] G. Buondonno, J. Carpentier, G. Saurel, N. Mansard, A. De Luca, and J. Laumond. Actuator design of compliant walkers via optimal control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 705–711, 2017.

[32] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane. Joint optimization of robot design and motion parameters using the implicit function theorem. In *RSS '17: XIII Robotics Science and Systems Conference*, volume 13, Cambridge, United States, 2017.

[33] A. Schulz, C. Sung, A. Spielberg, W. Zhao, R. Cheng, E. Grinspun, D. Rus, and W. Matusik. Interactive robogami: An end-to-end system for design of robots with ground locomotion. *The International Journal of Robotics Research*, 36(10):1131–1147, 2017.

[34] R. Desai, B. Li, Y. Yuan, and S. Coros. Interactive co-design of form and function for legged robots using the adjoint method. In *CLAWAR '18: 21th International Conference on Climbing and Walking Robots*, pages 125–133, Panama City, Panama, 2018.

[35] Y. Yesilevskiy, Z. Gan, and C. D. Remy. Energy-Optimal Hopping in Parallel and Series Elastic One-Dimensional Monopeds. *Journal of Mechanisms and Robotics*, 10(3), 2018.

[36] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter. Jointly learning to construct and control agents using deep reinforcement learning, 2018. [Online]. Available: *arXiv:1801.01432* [cs.RO].

[37] A. Giusti, M. J. A. Zeestraten, E. Icer, A. Pereira, D. G. Caldwell, S. Calinon, and M. Althoff. Flexible automation driven by demonstration: Leveraging strategies that simplify robotics. *IEEE Robotics and Automation Magazine*, 25(2):18–27, 2018.

[38] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane. Computational co-optimization of design parameters and motion trajectories for robotic systems. *The International Journal of Robotics Research*, 37(13-14):1521–1536, 2018.

[39] L. Carlone and C. Pinciroli. Robot co-design: Beyond the monotone case. In *ICRA '19: International Conference on Robotics and Automation*, pages 3024–3030, 2019.

[40] M. Pirron and D. Zufferey. Mperl: Hardware and software co-design for robotic manipulators. In *IROS '19: IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7784–7790, 2019.

[41] M. Chadwick, H. Kolvenbach, F. Dubois, H. F. Lau, and M. Hutter. Vitruvio: An open-source leg design optimization toolbox for walking robots. *IEEE Robotics and Automation Letters*, 5(4):6318–6325, 2020.

[42] T. Chen, Z. He, and M. Ciocarlie. Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning. In *CoRL '20: Conference on Robotic Learning*, Auckland, New Zealand, 2020.

[43] G. Fadini, T. Flayols, A. Del Prete, N. Mansard, and P. Souères. Computational design of energy-efficient legged robots: Optimizing for size and actuators. In *ICRA '21: IEEE International Conference on Robotics and Automation*, pages 9898–9904, Xi'an, China, 2021.

[44] T. Dinev, C. Mastalli, V. Ivan, S. Tonneau, and S. Vijayakumar. A versatile co-design approach for dynamic legged robots, 2022. [Online]. Available: *arXiv:2103.04660* [cs.RO].

[45] G. Fadini, T. Flayols, A. Del Prete, and P. Souères. Simulation aided co-design for robust robot optimization. *IEEE Robotics and Automation Letters*, 7(4):11306–11313, 2022.

[46] C. Schaff, A. Sedal, and M. R. Walter. Soft robots learn to crawl: Jointly optimizing design and control with sim-to-real transfer, 2022. [Online]. Available: *arXiv:2202.04575* [cs.RO].

[47] A. Belmonte-Baeza, J. Lee, G. Valsecchi, and M. Hutter. Meta reinforcement learning for optimal design of legged robots. *IEEE Robotics and Automation Letters*, 7(4):12134–12141, 2022.

[48] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, 2007.

[49] M. Hutter, C. Gehring, M. Bloesch, M.A. Hoepflinger, C.D. Remy, and R. Siegwart. Starleth a compliant quadrupedal robot for fast, efficient, and versatile locomotion. In *CLAWAR '12: 15th International Conference on Climbing and Walking Robots*, Maryland, USA, 2012.

[50] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 282–291, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[51] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *9th IFAC World Congress Proceedings Volumes*, volume 17, pages 1603–1608, Budapest, Hungary, 1984.

[52] P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[53] G. Bravo-Palacios, A. Del Prete, and P.M. Wensing. One robot for many tasks: Versatile co-design through stochastic programming. *IEEE Robotics and Automation Letters*, 5(2):1680–1687, 2020.

[54] © 2021 American Society of Mechanical Engineers ASME. Reprinted, with permission, from G. Bravo-Palacios, G. Grandesso, A. Del Prete, and P.M. Wensing. Robust Co-Design: Coupling Morphology and Feedback Design Through Stochastic Programming . *Journal of Dynamic Systems, Measurement, and Control*, 144(2), 11 2021. 021007.

[55] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control, 2020. [Online]. Available: *arXiv:1909.04947* [cs.RO].

[56] R.H. Byrd, J. Nocedal, and R.A. Waltz. *Knitro: An Integrated Package for Nonlinear Optimization*, pages 35–59. Springer US, Boston, MA, 2006.

[57] C. Sartore, L. Rapetti, and D. Pucci. Optimization of humanoid robot designs for human-robot ergonomic payload lifting. In *Humanoids '22: IEEE International Conference on Humanoid Robots*, pages 722–729, Ginowan, Japan, 2022.

[58] J. Andersson, J. Åkesson, and M. Diehl. Casadi: A symbolic package for automatic differentiation and optimal control. In *Recent advances in algorithmic differentiation*, pages 297–307. Springer, 2012.

[59] L. Natale, C. Bartolozzi, D. Pucci, A. Wykowska, and G. Metta. icub: The not-yet-finished story of building a robot child. *Science Robotics*, 2(13):eaaq1026, 2017.

[60] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016. [Online]. Available: *arXiv:1606.01540* [cs.LG].

[61] H. Cho, S. Oh, and D. Choi. A new evolutionary programming approach based on simulated annealing with local cooling schedule. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*, pages 598–602, Anchorage, Alaska, 1998.

[62] S. "Gao, S. Kong, and E.M." Clarke. dreal: An smt solver for nonlinear theories over the reals. In *Automated Deduction – CADE-24*, pages 208–214, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[63] L.-C.T. Wang and C.C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, 1991.

[64] A.W. Winkler, C.D. Bellicoso, M. Hutter, and J. Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

[65] Inc. The MathWorks. *Global Optimization Toolbox*. Natick, Massachusetts, USA, 2022.

[66] Inc. The MathWorks. *Parallel Computing Toolbox*. Natick, Massachusetts, USA, 2022.

[67] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Bloesch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer. Anymal - toward legged robots for harsh environments. *Advanced Robotics*, 31(17):918–931, 2017.

[68] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti. Crocoddyl: An efficient and versatile framework for multi-contact optimal control, 2019. [Online]. Available: *arXiv:1910.00093* [cs.RO].

[69] M.J.D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, 1994.

[70] M.J.A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D.G. Caldwell. An approach for imitation learning on riemannian manifolds. *IEEE Robotics and Automation Letters*, 2(3):1240–1247, 2017.

[71] E. Icer, A. Giusti, and M. Althoff. A task-driven algorithm for configuration synthesis of modular robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5203–5209, 2016.

[72] IBM ILOG Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[73] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, I. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms

for scientific computing in python. *Nature Methods*, 17:261–272, 2020.

[74] A. De Luca and L. Ferrajoli. A modified newton-euler method for dynamic computations in robot fault detection and control. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3359–3364, 2009.

[75] E. Icer, A. Giusti, and M. Althoff. task-driven algorithm for configuration synthesis of modular robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5203–5209, 2016.

[76] J. Watson, D.L. Woodruff, and W.E. Hart. Pysp: modeling and solving stochastic programs in python. *Mathematical Programming Computation*, 4(2):109–149, 2012.

[77] Inc. COIN-OR Foundation. *Global Optimization Toolbox*. State of Maryland, USA.

[78] G.A. Pratt and M.M. Williamson. Series elastic actuators. In *1995 IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 399–406, Pittsburgh, Pennsylvania, 1995.

[79] C. Lee, S. Kwak, J. Kwak, and S. Oh. Generalization of series elastic actuator configurations and dynamic behavior comparison. *Actuators*, 6:1–26, 2017.

[80] S. Seok, A. Wang, D. Otten, and S. Kim. Actuator design for high force proprioceptive control in fast legged locomotion. In *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1970–1975, Vilamoura, Portugal, 2012.

[81] P.M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim. Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *IEEE Transactions on Robotics*, 33:509–522, 2017.

[82] G. Kenneally, A. De, and D.E. Koditschek. Design principles for a family of direct-drive legged robots. *IEEE Robotics and Automation Letters*, 1:900–907, 2016.

[83] J.B. Morrell and J.K. Salisbury. Parallel-coupled micro-macro actuators. *International Journal of Robotics Research*, 17:773–791, 1998.

[84] M. Zinn, B. Roth, O. Khatib, and J.K. Salisbury. A new actuation approach for human friendly robot design. *International Journal of Robotics Research*, 23:379–398, 2004.

[85] K.Y. Toumi H. Asada. *Direct-Drive Robots, Theory and Practice.* MIT Press, Cambridge, Massachusetts, 1987.

[86] A.G. Leal Junior, R.M. de Andrade, and A.B. Filho. Series elastic actuator: Design, analysis and comparison. *Recent Advances in Robotic Systems*, 2016.

[87] S. Seok, A. Wang, M.Y. Chuah, D.J. Hyun, J. Lee, D.M. Otten, J.H. Lang, and S. Kim. Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot. *IEEE/ASME Transactions on Mechatronics*, 20(3):1117–1129, 2015.

[88] W.E. Hart, J. Watson, and D.L. Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[89] Hsl: A collection of fortran codes for large scale scientific computation. See URL `http://www.hsl.rl.ac.uk`, 2013.

[90] N. Kau, A. Schultz, N. Ferrante, and P. Slade. Stanford doggo: An open-source, quasi-direct-drive quadruped. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 6309–6315, Montreal, Canada, 2019.

[91] Maxon Group. High precision drives and systems 2019-2020 catalog.

[92] A. Babarit and A.H. Clement. Optimal latching control of a wave energy device in regular and irregular waves. *Appl. Ocean Res.*, 28:77–91, 2006.

[93] F. Saupe, J.C. Gilloteaux, P. Bozonnet, Y. Creff, and P. Tona. Latching control strategies for a heaving buoy wave energy generator in a random sea. In *19th IFAC*, pages 7710–7716, Cape Town, South Africa, 2014.

[94] K. Budal and J. Falnes. Interacting point absorbers with controlled motion. In B.M. Count, editor, *Power from Sea Waves*, pages 381–399. Academic Press, London, 1980.

[95] H. Dai and R. Tedrake. Optimizing robust limit cycles for legged locomotion on unknown terrain. In *51st IEEE Conf. Decis. Control.*, pages 1207–1213, December 2012.

[96] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[97] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[98] X. Li, A. Tomasgard, and P.I. Barton. Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs. *Journal of optimization theory and applications*, 151(3):425, 2011.

[99] J. Di Carlo B. Katz and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *IEEE Int. Conf. Robot. Autom.*, pages 6295–6301, 2019.

[100] M. Neunert et al. Whole-body nonlinear model predictive control through contacts for quadruped. *IEEE RA-L*, 3(3):1458–1465, 2018.

[101] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov. An integrated system for real-time model predictive control of humanoid robots. In *IEEE-RAS Humanoids*, pages 292–299, 2013.

[102] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo. Mpc for humanoid gait generation: Stability and feasibility. *IEEE RA-L*, 36(4):1171–1188, 2020.

[103] N. Mansard, A. Del Prete, M. Geisert, S. Tonneau, and O. Stasse. Using a memory of motion to efficiently warm-start a nonlinear predictive controller. In *IEEE International Conference on Robotics and Automation*, pages 2986–2993, 2018.

[104] M. Bardi and I. Capuozzo Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, volume 12. Birkhäuser, 1997.

[105] J.N. Tsitsiklis. Globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[106] L.C. Polymenakos, D.P. Bertsekas, and J.N. Tsitsiklis. Implementation of efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 43(2):1528–1538, 1998.

[107] A. Gorodetsky, S. Karaman, and Y. Marzouk. Efficient high-dimensional stochastic optimal motion control using tensor-train decomposition. In *Proceedings of Robotics: Science and Systems*, 2015.

[108] E. Stefansson and Y.P. Leong. Sequential alternating least squares for solving high dimensional linear hamilton-jacobi-bellman equation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3757–3764, 2016.

[109] Y. Hou, H. Hong, Z. Sun, D. Xu, and Z. Zeng. The control method of twin delayed deep deterministic policy gradient with rebirth mechanism to multi-dof manipulator. *Electronics*, 10(7), 2021.

[110] Z. Zhang, J. Chen, Z. Chen, and W. Li. Asynchronous episodic deep deterministic policy gradient: Toward continuous control in computationally complex environments. *IEEE Transactions on Cybernetics*, 51(2):604–613, 2021.

[111] S. Levine and V. Koltun. Guided policy search. In *Proceedings of the International Conference on Machine Learning*, pages III–1–III–9, 2013.

[112] I. Mordatch and E. Todorov. Combining the benefits of function approximation and trajectory optimization. In *Proceedings of Robotics: Science and Systems*, volume 4, 2014.

[113] M. Vecerik et al. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards, 2017. [Online]. Available: *arXiv:1707.08817* [cs.LG].

[114] A. Filos et al. Psiphi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. In *Proceedings of the International Conference on Machine Learning*, pages 3305–3317, 2021.

[115] T. Osa, A.M.G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann. Guiding trajectory optimization by demonstrated distributions. *IEEE RA-L*, 2(2):819–826, 2017.

[116] A.S. Morgan, D. Nandha, G. Chalvatzaki, C. D'Eramo, A.M. Dollar, and J. Peters. Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6672–6678, 2021.

[117] D.P. Bertsekas. *Reinforcement Learning and Optimal Control*, pages 58–69. Athena Scientific, first edition, 2019.

[118] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis. Guiding trajectory optimization by demonstrated distributions. *Commun. Comput. Phys.*, 28(5):1671–1706, 2020.

[119] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2015. [Online]. Available: *arXiv:1412.6980v9* [cs.LG].

[120] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*, pages 76–80. The MIT Press, second edition, 2018.

[121] L.E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[122] W.M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu. Sobolev training for neural networks. In *Proc. of '17 NeurIPS*, volume 30, 2017.

[123] F. Farshidian E. Jelavic and M. Hutter. Combined sampling and optimization based planning for legged-wheeled robots. In *Proc. of '21 IEEE ICRA*, pages 8366–8372, 2021.

[124] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay, 2018. [Online]. Available: *arXiv:1707.01495* [cs.LG].

[125] J.C. Duchi, P.L. Bartlett, and M.J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

[126] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R.M.Sumner, W. Matusik, and B. Bickel. Computational design

of mechanical characters. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.