Exploration des descriptions d'état et des simulations de robots souples pour l'apprentissage et le contrôle

Etienne Ménager

21 Décembre 2023
Version finale

Exploring state descriptions and soft robot simulations for learning and control.

Etienne Ménager

December 21st, 2023
Version: Final Draft

Université de Lille - MADIS

Centre Inria de l'Université de Lille
DEFROST (Deformable Robotic Software)

Thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

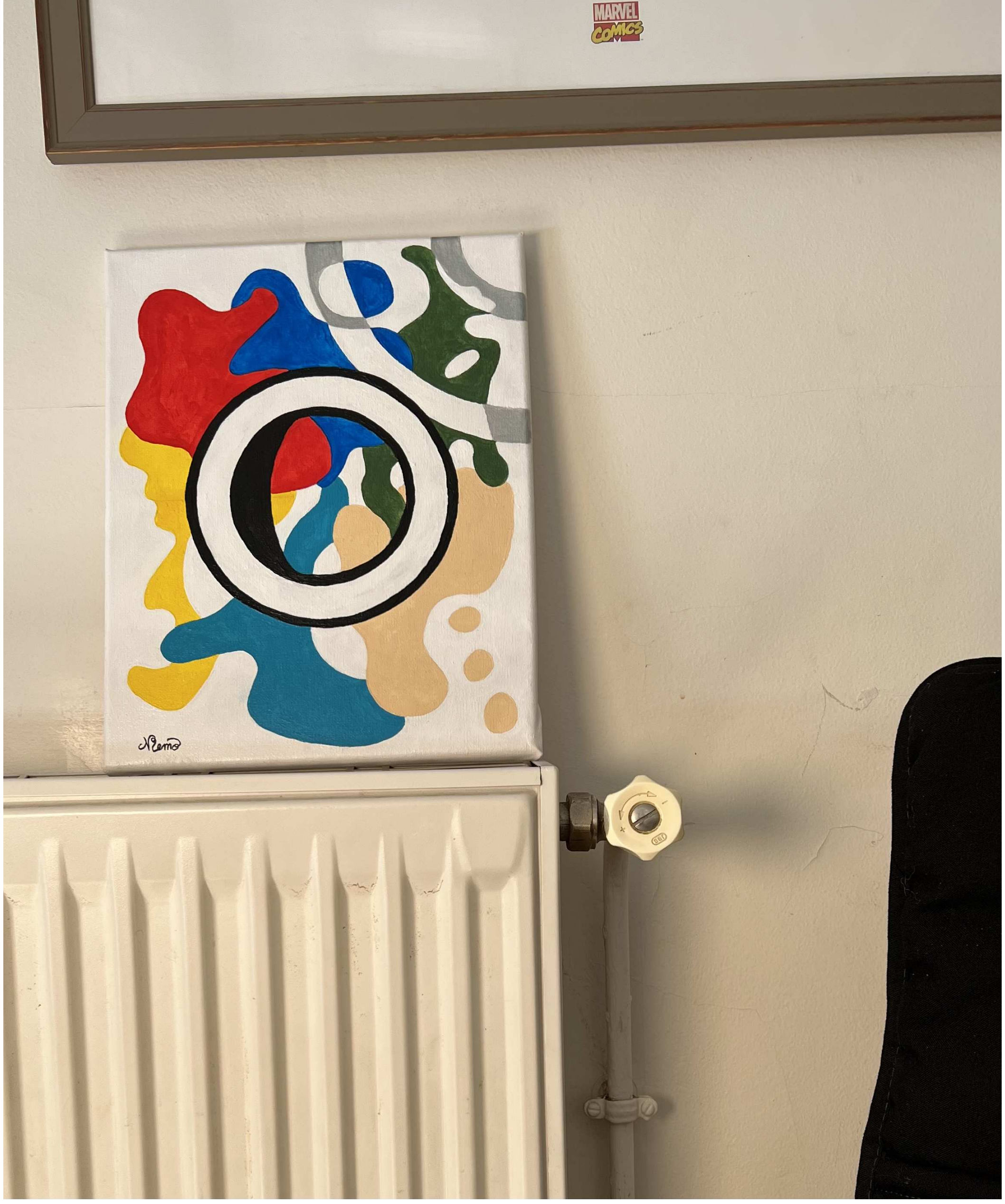
Exploring state descriptions and soft robot simulations for learning and control.

Etienne Ménager

December 21st, 2023

Etienne Ménager
Exploring state descriptions and soft robot simulations for learning and control.
Thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science, December 21st, 2023
Reviewers: Vincent Padois and Kanty Rabenorosoa
Supervisor: Christian Duriez

Université de Lille - MADIS
DEFROST (Deformable Robotic Software)
Centre Inria de l'Université de Lille



> *1. Émotions.*
>
> — **Nemo**

During the three years of my PhD, I dedicated time to painting. I believe that knowledge can be obtained in different ways, whether through science or art. I painted life. Not necessarily my own life but probably a little. I would like to share some of my paintings with you as both a journey through life and through this work. The collection is titled "De l'âme", which references the human being represented by a double circle, a soul inside a body. The first painting represents the self, subject to the influence of emotions. According to American psychologist Paul Ekman, there are six universal primary emotions represented by six different colors in the painting. Just as in life, working on a PhD leads to experiencing all of these emotions, such as the joy of publishing an article in a well-known journal or the anger of not achieving the desired results. It is only in the contemplation of the universal that emotions vanish.

Abstract

Soft robotics is a subfield of robotics where soft materials are used to create robots that can adapt to their environment. This adaptability allows for numerous innovations, such as in medical robotics and the manipulation of delicate objects, but also presents challenges in areas like modeling, control, and design. The use of Finite Element Methods (FEM) enables accurate modeling and simulation of soft robots. It enables both the evaluation of soft robot designs prior to manufacturing and their control. The DEFROST team at Inria Lille has developed several tools for controlling soft robots, including the joint use of inverse FEM modelling and convex optimisation. However, these methods were developed for low-level control over a short time horizon. The objective of this PhD is to extend this low-level control to the control of soft robots over a task-time horizon.

In this work, various simulation and machine learning tools will be used to find actuation sequences for solving tasks. Although these tools are powerful, they each have limitations. For instance, learning-based control is effective in tackling complex tasks but requires a significant amount of data for learning. FEM models can simulate a wide range of behaviors but the simulation process can be slow. One of the common problems of these different tools is the size of the systems to be solved and therefore the representation of the soft robot state. Depending on the discipline (robotics, machine learning, simulation), the notion of state does not have the same definition and does not have the same purpose. As the number of parameters increases with the time horizon of the task to perform, a trade-off must be found between the different representations in order to efficiently formulate the optimisation problems. This research focuses on adapting the description of the state of soft robots to optimise their actuation variables on the task-time horizon. This is important because the choice of model determines what is actually possible to do in terms of learning, control and even design.

Résumé

La robotique déformable est un sous-domaine de la robotique où des matériaux souples sont utilisés pour créer des robots capables de s'adapter à leur environnement. Cette adaptabilité permet de nombreuses innovations, que ce soit dans le domaine médicale ou la manipulation d'objets délicats, mais présente également de nombreux défis dans les domaines de la modélisation, du contrôle et de la conception. La méthode des éléments finis (FEM) permet une modélisation et une simulation précises des robots souples. Elle permet à la fois de les contrôler, mais aussi d'évaluer leurs conceptions avant de les fabriquer. L'équipe DEFROST du Centre Inria de l'Université de Lille a développé plusieurs outils pour contrôler les robots souples, y compris l'utilisation conjointe de la modélisation inverse FEM et de l'optimisation convexe. Cependant, ces méthodes ont été développées pour un contrôle bas niveau sur un horizon temporel court. L'objectif de ce doctorat est d'étendre ce contrôle bas niveau au contrôle de robots souple sur un horizon de temps relatif à l'exécution d'une tâche.

Dans ce travail, divers outils de simulation et d'apprentissage automatique seront utilisés pour trouver des séquences d'actionnement permettant de résoudre des tâches. Bien que ces outils soient puissants, ils ont tous des limites. Par exemple, la commande basée sur l'apprentissage est efficace pour résoudre des tâches complexes, mais elle nécessite une quantité importante de données pour l'apprentissage. Les modèles FEM peuvent simuler un large éventail de comportements, mais le processus de simulation peut être lent. L'un des problèmes communs à ces différents outils est la taille des systèmes mathématiques à résoudre et donc la représentation de l'état du robot souple. Selon la discipline (robotique, apprentissage automatique, simulation), la notion d'état n'a pas la même définition et n'a pas la même finalité. Le nombre de paramètres augmentant avec l'horizon temporel de la tâche à réaliser, un compromis doit être trouvé entre les différentes représentations afin de formuler efficacement les problèmes d'optimisation. Ce travail de recherche se concentre sur l'adaptation de la description de l'état des robots souples afin d'optimiser leur actionnement sur l'horizon temporel de la tâche. Ceci est important car le choix du modèle détermine ce qu'il est réellement possible de faire en termes d'apprentissage, de contrôle et même de conception.

Contents

1 Overall Introduction

Contents

2. Présent.

— Nemo

In his essay "Essai sur les données immédiates de la conscience", Bergson distinguishes between duration and time. Time is a measurement of repetition in space, whereas duration presupposes something that ensures continuity between successive states. These moments follow each other. The past influences the future through memory, while the future influences the past in anticipation of possibilities. The perception of duration as a continuity of passing moments is possible only through elevation. Through this elevation, a particular moment, the present, becomes clear. This is the moment of possibility, at which everything can begin. This is where we begin.

The limits of my language mean the limits of my world.

— **Ludwig Wittgenstein**
Tractatus logico-philosophicus, 1921

The objective of this general introduction is to provide context for this study and to emphasize its contributions. First, we will introduce the field of soft robotics and explore its potential and limitations. Then, we will explain the significance of studying the state description of soft robots for control and design tasks. Finally, we will highlight the key contributions of this PhD study.

1.1 What is a soft robot?

1.1.1 From rigid to soft robotics

Robotics has first revolutionized industries by enabling automation of difficult and repetitive tasks. However, robots have since spread beyond factories and can now be found in various sectors, such as the medical field where surgical robots help during minimally invasive surgical procedures, and in human habitats with personal assistance or domestic robots. For a long time, these robots were constructed with rigid components, making them easy to design, model, and control. This rigidity gives the robot great precision, but can lead to dangerous interactions with fragile environments, such as breakable objects, humans or organs.

To address this issue, researchers have investigated the potential of soft robots. Soft robots are built using soft materials that provide intrinsic flexibility. This allows them to adapt to their surroundings [MC16], to deform and interact with complex and uncertain environments, and to perform tasks requiring flexibility and dexterity. The first soft gripper was developed in 1991 [SIT91]. In the following decade, various designs of soft robots have emerged, primarily inspired by nature. Currently, automatic design tools are used to design these robots [SSW22]. Examples of soft robots are shown in Figure 1.1. Soft robotics has a wide range of applications which meet challenges in robotics: for instance, moving in confined spaces [RDM19], gripping of fragile objects [Sin+19], interacting with living or biological components [Cia+18] and solving complex tasks using contacts with the environment [Coe19].

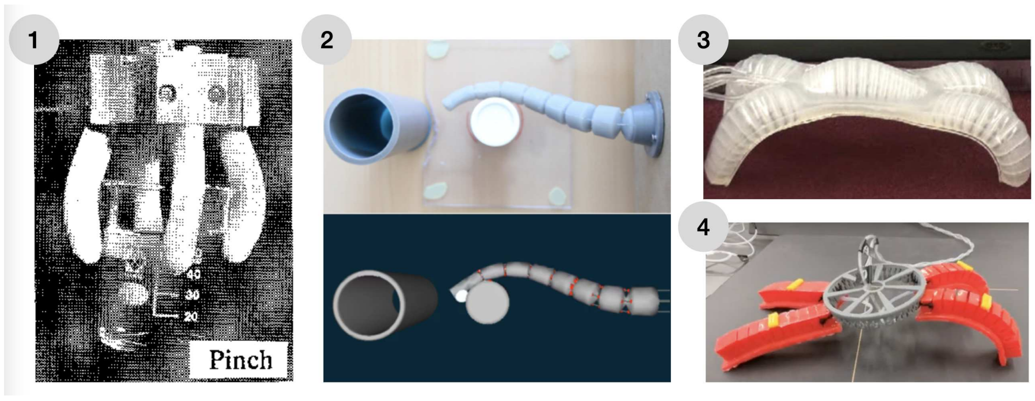


Fig. 1.1: Four examples of soft robots, including: 1) The first soft gripper introduced in [SIT91]. 2) A robot inspired by an elephant trunk [CED17], simulated (bottom), and manufactured (top). 3) A pneumatic crawling robot presented in [She+11]. 4) The automatic design of a soft robot presented in [SSW22].

1.1.2 Formal definition of a soft robot

There is a multitude of soft robots made of different materials (silicone rubber, biomaterials, polymers, etc.), actuation strategies (cable, pneumatic, electric, etc.), and geometries. However, we can propose a general definition of soft robots from these diverse examples.

Definition 1.1.1 (Soft robot). We consider that a soft robot is a robot whose movements are induced by deformations, either by actuation or by an external force. The deformations are sufficient to cause significant changes in position and orientation, and lead to non-linear behaviour.

This definition implies several properties, depending on how the robot is modeled. For example, the study of deformation via the mechanics of continuous media is based on the assumption that the structure has an infinite number of Degrees of Freedom (DoFs), that is an infinite number of possible independent relative motions. This leads to an under-actuated structure (which does not have one actuation for each DoF), with possible redundant actuators and several possible configurations to achieve the same goal by deformation.

1.2 State description for soft robotics

1.2.1 Description of the soft robot state

According to current knowledge, at the smallest scale of reality, a robot is composed of a large number of particles. We call this set of particles the complete state of the robot. This is a state that is inaccessible in practice, an ideal state in the Platonic sense. Describing the robot's behaviour at this level, that is predicting precisely everything that happens at the atomic level, is unrealistic. A modelling stage is therefore necessary. To describe the behaviour of a soft robot, modelling consists in representing its complete states by simplified and chosen states.

The notion of state is not universal and depends on the goal or the behaviour to be modelled. The mechanics of continuous media describes the state of a soft robot with an infinite number of DoFs. The assumption is that everything is continuous in order to write the partial derivatives of motion. The finite element approach represents the state of the soft robot using a mesh, that is a finite set of points. This mesh is chosen in order to converge towards the solutions of continuum mechanics while being small enough to be easily calculated. The starting point is therefore the continuous state of the mechanics of continuous media, which is then discretised. A roboticist is interested in the robot's input-output relation, and will therefore model the robot from the point of view of its actuators and effectors. The use of optimisation tools such as Reinforcement Learning (RL) involves choosing a robot state that captures its behaviour over time. Model reduction methods can also be used to obtain compact representations of the robot's state.

Each approach seeks to describe the complete state of the robot through a choice of model and parameterisation. In practice, this choice is made in order to capture information about the configurations of the robot and its static, kinematic or dynamic deformation behaviour. In this PhD thesis we will focus on different levels of robot modelling in order to control the robot at the task-time horizon.

1.2.2 Significance of soft robots' state description

Soft robots deform to interact with their environment, and this deformation must be considered in their modeling, control, and design. The state of a soft robot, and thus its body, plays a crucial role in these fields. We have to deal with embodiment, even in the control of the robot, in the sense that the body and the "mind" are inseparable [SG05]. However, describing the body is a challenge, particularly in soft robotics. A compact description of soft robots is required, while containing sufficient information to depict the behavior of the robot either in terms of deformation or interaction with the environment. For example, for a manipulator, it is crucial to consider the contacts with the object to be manipulated in the description of the robot because they determine its final shape. However, an extremely dense representation is undesirable because it leads to difficult parameterisation and expensive computations.

The compact and expressive description of the body of a soft robot remains an open question. Various methods exist based on different description spaces, such as the actuation space (e.g., the pressure in a pneumatic actuator or the torque on a joint), joint space (e.g., the length of a cable or the angle at a joint), configuration space (e.g., the deformations of the robot or the position of the different points of the robot), or task space (e.g., the position of the end effector of a manipulator or the average velocity of a moving robot). These different spaces are shown in Figure 1.2. Other spaces, such as mechanical constraint spaces or abstract learned spaces, can also be considered to describe the robot.

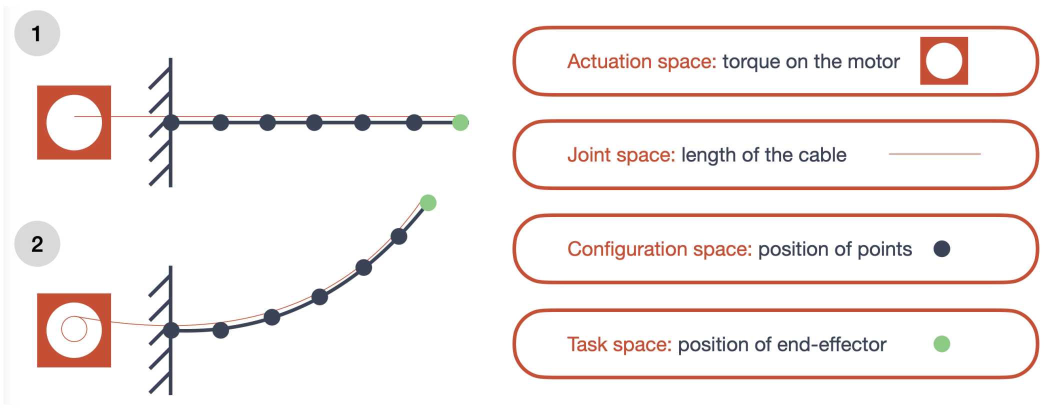

1
2
Actuation space: torque on the motor
Joint space: length of the cable
Configuration space: position of points
Task space: position of end-effector

Fig. 1.2: Illustration of different spaces that can be used to describe the state of a robot. A trunk-like robot (represented by a dark gray segment) fixed at one end is driven by a cable (represented by a thin red segment) pulled by a motor (represented by a red square with a white circle inside). These elements, as well as particular points of the robot (in dark gray) or the end-effector (in green) can be used to describe the state of the robot. 1) Robot at rest. 2) Robot in the actuated state.

The modelling of soft robots and the choice of parameterisation entirely determine the tools that can be used for control and design. Some tools rely on a particular description: control using convex optimisation requires the position of the effector to be described in a convex space; RL uses a state observation that must capture information about both the robot and its environment; optimisation of a soft robot design relies on a parametric description of the robot. Therefore, multiple tools are available but with a lack of connections between them. Establishing these links is challenging because of the different assumptions each tool implies regarding the state of the robot.

To reach the objective of this PhD, we will establish methods that combine various modeling, control, and design tools. We hypothesize that these methods can be developed by exploring the description of the state of a soft robot. This approach is supported by the quote at the beginning of this chapter. The method of modeling, controlling, and designing a soft robot depends on the description of its state and the language used to do so. Furthermore, the limitations of the tools ("the limits of my world") are often caused by the limitations of the language used to describe the state of the soft robot ("the limits of my language").

1.3 Conclusion

The significance of the robot state description is emphasized in this introduction. The study of modeling in soft robotic is important because it determines the behavior of the robot that can be represented, the way it is controlled, or the shapes that can be optimized.

In this manuscript, we propose an exploratory work on the modeling and simulation of soft robots, mainly for their control. The physics of these systems is highly non-linear and complex, and leads to the description of robot states with high-dimensional mathematical quantities. As we will see later in this manuscript, the current control method based on these quantities are local, in the sense that the robot's behavior is optimized on a single targeted configuration. The result is a low-level actuation command with a short time horizon. On the other hand, optimizing the behavior of the robot on several successive configurations, that is over a trajectory, would lead to advanced and complex behaviors. This requires to consider the global task execution through « high-level » planning. The general idea is to add control methods on top of low-level control in order to generate a sequence of target configurations to solve a more complex task. This could be a sequence of movements to achieve locomotion in a complex environment (catheter in arteries, legged robots, etc.), or to enable grasping and manipulation of an object (gripper, trunk, etc.). These examples show that such control takes place over a longer time horizon, which is why we speak of long time horizon tasks.

However, the dimensionality of the system limits the development of theses methods. In this context, the problem can be reformulate in two points. It consists, on the one hand, to develop models and associated state descriptions enabling computational parsimony while guaranteeing a form of fidelity to the underlying system physics. On the other hand, it concerns the use of theses simplified models in combination with automatic learning methods for the generation of complex control policies, inaccessible to local model-based control methods which solve constrained optimisation problem at each time step. We will see that, if these descriptions of the state are used initially for control purposes, some can be used for design. We propose a solution to this problem by exploring the use of different description methods for soft robots, and by changing or combining description spaces to facilitate the use of existing tools that are not easily combined.

This work is based on the use of digital robot twins, modeled and simulated using FEM simulators. This choice is based on the expertise of the DEFROST team and the existence of realistic robot models that have already been used to perform control transfers to physical models [CED17; GD18]. The underlying assumption is that the control law found for the simulated model can be transferred to the corresponding physical model. Therefore, simulation is the ground truth and serves as the foundation for the tools developed in this PhD thesis.

1.4 Contribution and structure of the thesis

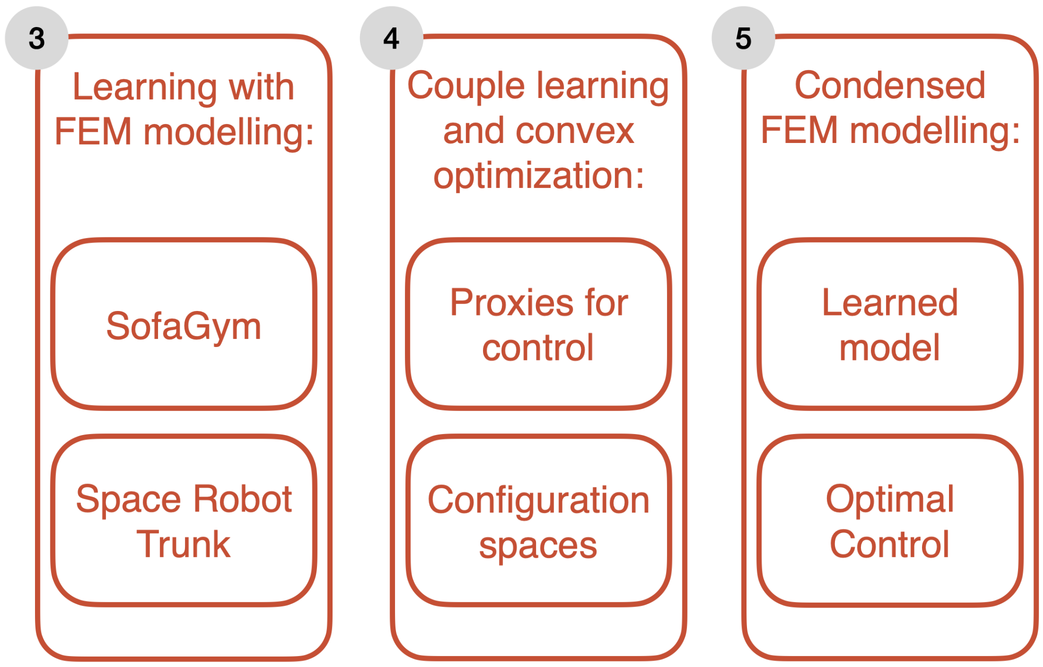


Fig. 1.3: Contributions of this PhD thesis. Chapter 3: Using FEM simulation for learning. Exploring control methods using the SofaGym API and applying a RL algorithm with a trunk-like robot in space. Chapter 4: Coupling learning-based and convex-optimisation-based control methods. Representation of the robot in the configuration space with the robot proxy and splitting the observation space with respect to the contact configuration of the robot. Chapter 5: Using condensed FEM model. Learning a reduced mechanical model for control and design applications.

This work presents the development of tools to identify the limitations of current control approaches in soft robotics, proposes solutions to overcome these limitations based on different model and parametrisation of the robot, and explores how some of these solutions can be extended to address design issues. The key contributions of this study are as follows:

- Chapter 3: Development of an API that combines machine learning algorithms with FEM simulations of soft robots. This interface enables the learning of control strategies for any FEM model and allows for exploring the limitations of existing tools such as optimisation-based and learning-based control in soft robotics.

- Chapter 4: Development of two description spaces to efficiently couple learning-based and optimisation-based control. The first is a reduced mechanical model of a soft robot that allows to describe it in configuration space. This model enables efficient learning of a control strategy in the configuration space, which can then be transferred to the robot using optimisation tools. The second is a description of the state of the robot as a set of contact configuration spaces. In addition to improving the sample efficiency, this method allows the integration of prior knowledge about the robot behavior into the learning process and the choice of when to use learning control and optimisation control.

- Chapter 5: Exploitation of a compact mechanical model of the robot based on the description of the robot in the constraint space (effector, actuator, contact). This model can be learned to control the robot and design it. This description of the robot's state can also be used in an optimal control framework to optimize trajectories.

The SofaGym API was developed in collaboration with Pierre Schegg. Research on condensed model learning was carried out in collaboration with Tanguy Navez. The various contributions presented in Figure 1.3 are explored and expanded in the remaining sections of this manuscript.

These topics have been covered in multiple scientific publications, conferences, and journals as either the principal author or co-author:

- Pierre Schegg, Etienne Ménager, Elie Khairallah, Damien Marchal, Jérémie Dequidt, Philippe Preux, and Christian Duriez, "*SofaGym: An Open Platform for Reinforcement Learning Based on Soft Robot Simulations*", Soft Robotics, 2022.

- Etienne Ménager, Tanguy Navez, Olivier Goury and Christian Duriez, "*Direct and inverse modeling of soft robots by learning a condensed FEM model*", ICRA, 2023.

- Camilla Agabiti, Etienne Ménager and Egidio Falotico, "*Whole-arm Grasping Strategy for Soft Arms to Capture Space Debris*", Robosoft, 2023.

- Etienne Ménager, Quentin Peyron and Christian Duriez, "*Toward the use of proxies for efficient learning manipulation and locomotion strategies on soft robots*", RAL, accepted.

- Etienne Ménager and Christian Duriez, "*Learning control strategy in soft robotics through a set of configuration spaces*", to be submitted.

- Etienne Ménager, Alexandre Bilger, Wilson Jallet, Justin Carpentier and Christian Duriez, "*Condensed semi-explicit dynamics for optimal control in soft robotics*", to be submitted.

- Tanguy Navez, Etienne Ménager, Paul Chaillou, Olivier Goury and Christian Duriez, "*Modeling, Embedded Control and Design of Soft Robots using a Learned Condensed FEM Model*', to be submitted.

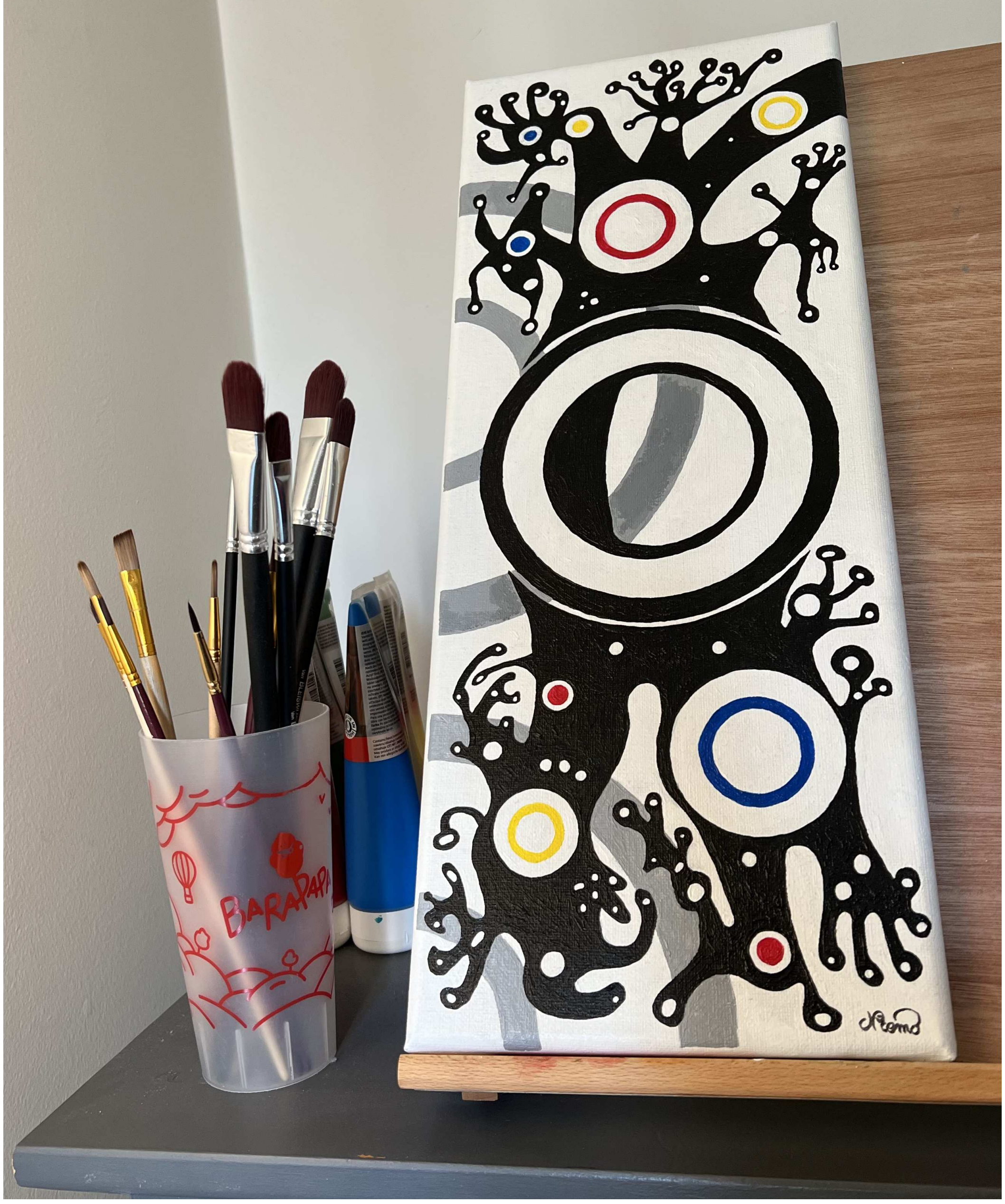
Each project relies on the creation of open-source code, which is hosted on my git repository. Specifically, I have developed a plugin for SOFA, "SofaGym: An OpenAI Gym API for SOFA Simulations" (Github, 2020).

1.5 Acknowledgments

I would like to express my gratitude to my PhD supervisor, Christian Duriez, for his unwavering trust and support in exploring various subjects. He guided me throughout my projects, and I hope we will continue to collaborate in the future. I am also grateful to the DEFROST team, particularly Pierre Schegg, Yinoussa Adagolodjo and Alexandre Bilger who welcomed me and allowed me to work on several projects with them. A special mention for my deskmate, Tanguy Navez, with whom I had the pleasure of working and sharing many hours to redesign the world of soft robotics. I also thank Paul Chaillou, who traveled with me to the other side of the world for my first conference. Thank you to Quentin Peyron, for the work we did together, the good mood and the attentive review of this manuscript. Finally, I thank Justin Carpentier for allowing me to develop certain aspects of this study.

2 Related Work

Contents



3. Connexion.

— **Nemo**

Most stories are shared experiences. We build our identity from the interactions we have in society, with our loved ones, with the system, with our peers. We are the product of a complex web of colliding histories. This is evident in scientific research. As Isaac Newton said: "If I have seen further, it is by standing on the shoulders of giants". It is through building upon existing work that we can advance science.

The objective of this chapter is to introduce various mathematical tools and models that can be used for modelling, control, and design in soft robotics. The chapter begins by discussing different approaches for modelling a soft robot (2.1) and the choice of using FEMs in this study is discussed. Next, the chapter covers mathematical tools for control, including sequential decision problems (2.2.2) and optimisation based on mechanical modelling (2.2.3). Section 2.3 describes the design approaches in soft robotics. The chapter then discusses existing methods for decribing the state of a robot, such as the general framework of state learning and use of reduced models in humanoid robotics (2.4). Finally, the chapter outlines the research questions addressed in the next chapters with respect to the background (2.5). The goal is to use the current state-of-the-art to demonstrate the relevance and positioning of the scientific work presented in this PhD thesis. For each section, we highlight the link between the section and the rest of the manuscript in a grey square. Concepts and definitions that will be reused frequently are listed as "Useful concepts".

2.1 Modelling and simulation in soft robotics

2.1.1 Geometric, mechanical and learning-based approach

In robotics, a model links the robot's actuation a to its state description X. There are two main types of models: direct and inverse models. Direct models[1] are defined using the equation $X = f(a)$ where f is a function used to calculate X based on the robot actuation. These models are used for interactive simulations or for evaluating the evolution of robots over time. Inverse models, on the other hand, are defined using the equation $a = f^{-1}(X)$. Inverse models are used for control and path/motion planning in soft robotics, or for evaluating forces in a soft structure based on a given description X [Nav+20]. Modelling allows to determine how the robot's state description evolves based on the constraints imposed on it, or how the constraints evolve based on the robot's state description. Because modelling is expressed as a function of the robot's description, each description space leads to a different model.

Because of the nature of soft robots, it is in most cases impossible to develop an analytical model for them. Soft robots have many degrees of freedom that are not directly controlled, and their kinematics are influenced not only by geometry, but also by material properties and the eventual interactions with the environment

[1]In robotics, a direct model classically links the joint space to the operational space, which depends on the task. In the following, we use the term direct model as the relationship between actuators and states, to avoid this dependency.

through contacts. In the early 2010s, when the soft-robotics community experienced rapid growth, modelling and controlling soft robots were major challenges with no available solutions [Tri+08; Lip14; Maj13]. Since then, various approaches have been proposed to obtain the direct and inverse models of soft robots. These can be categorized into three categories: geometrical methods, mechanical models, and learning-based approaches. Examples are shown in Figure 2.1. Each of these models has its own advantages and disadvantages, and they all rely on a particular description of the state of the robot.

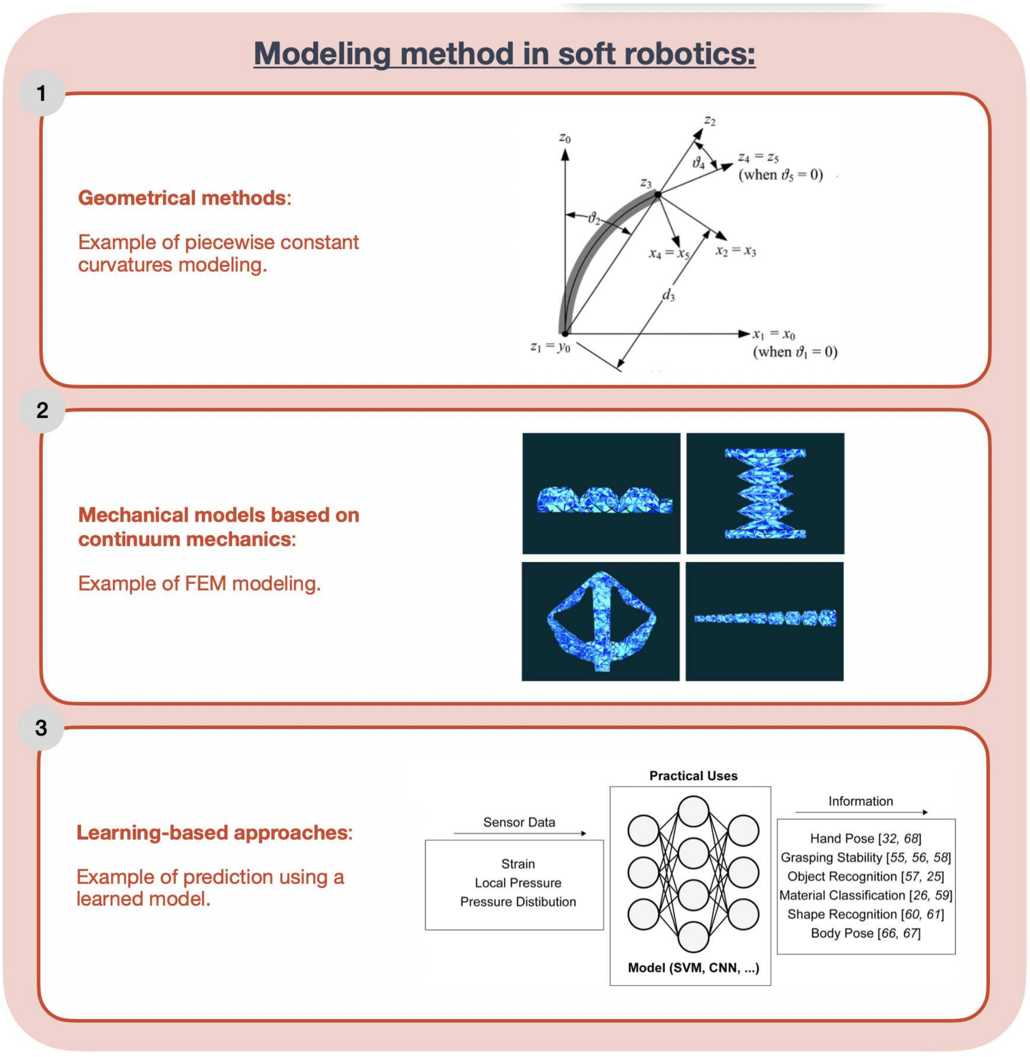

Modeling method in soft robotics:
1
Geometrical methods:
Example of piecewise constant curvatures modeling.
z_0
z_2
ϑ_4
$z_4 = z_5$ (when $\vartheta_5 = 0$)
z_3
ϑ_2
$x_4 = x_5$
$x_2 = x_3$
d_3
$z_1 = y_0$
$x_1 = x_0$ (when $\vartheta_1 = 0$)
2
Mechanical models based on continuum mechanics:
Example of FEM modeling.
3
Learning-based approaches:
Example of prediction using a learned model.
Sensor Data
Strain
Local Pressure
Pressure Distibution
Practical Uses
Model (SVM, CNN, ...)
Information
Hand Pose [32, 68]
Grasping Stability [55, 56, 58]
Object Recognition [57, 25]
Material Classification [26, 59]
Shape Recognition [60, 61]
Body Pose [66, 67]

Fig. 2.1: Examples of soft-robotic modelling using: 1) a geometric method, image from [WJ10]. 2) a mechanical model, image from [Coe+17]. 3) a learning-based approach, image from [Kim+21a].

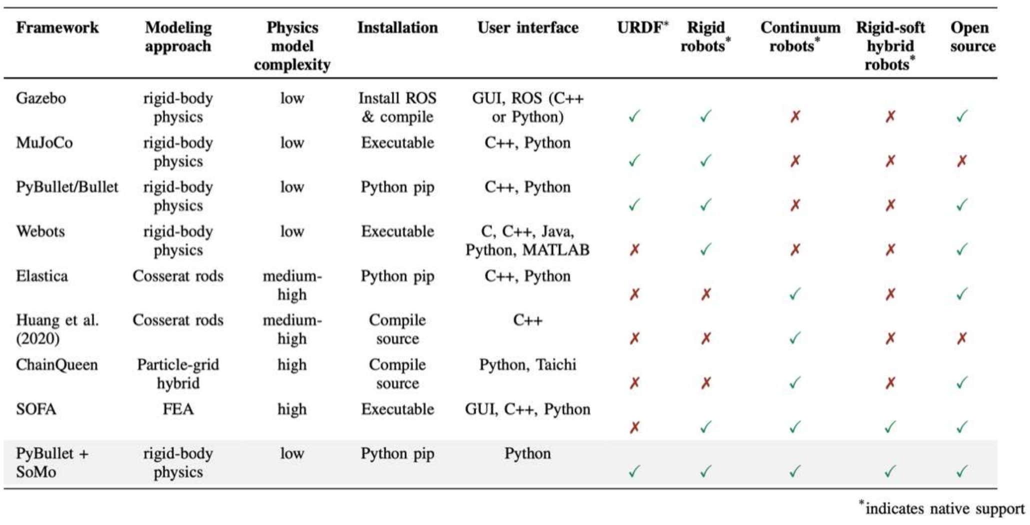
Geometrical methods rely on strong assumptions regarding the robot behavior, in particular that they must have a slender body composed of sections with constant curvature [WJ10]. These methods are limited to particular robots and the assumption becomes invalid when external forces (gravity, contact force) are applied, leading to a non-constant curvature. Mechanical models are based on continuum mechanics and can take different forms. They can be expressed as systems of partial differential equations with one spatial dimension, such as Euler-Bernoulli beams [SGW15] or Cosserat beams [Ren+17; Ant73], or more generally sets of non-linear equations such as FEM [Dur13]. These approaches are very general, but are limited by the fact that they require good experience in numerical modelling to be used effectively. In addition, these methods require a lot of computing resources, especially to be run in real-time. Finally, machine learning can be used for soft robotics modelling [Kim+21a]. The idea behind these approaches is to learn the function f according to $f(a;\theta) = X$ where θ are parameters that can be learned. These methods applies also to inverse modelling. They create a link between an input and output without an explicit mathematical formulation of the problem. During the inference phase, for example in direct modelling, θ is used to predict X as a function of a with high prediction speed. Control methods (trajectory optimization, closed loop control, etc.) can be developed based on this learning [Thu+19; Geo+16; Thu+17]. The counterpart is that the learned model is used as a black box, requires a very important dataset during the learning phase and in new situations, such as new contact configurations that are not present in the dataset, there is no guarantee that the model can generalise.

2.1.2 Soft robot simulators

The creation of autonomous systems presents many challenges for both design and control. Unlike physical prototypes, soft robot simulation is a safe, fast, and cost-effective way to test designs and control methods [Cho+20]. The virtual environments allow to test and validate soft robots's behavior under various simulation conditions (contact conditions, extreme configurations of the robot, and the presence of external objects). To be useful, the simulation must be realistic enough to accurately depict the behavior of the soft robot and its interactions. However, the more accurate the simulation, the more resources are required for the calculation, in particular due to the non-linearity of the underlying models. For example, FEM simulation, which has been tested and validated in the medical field and used for the real-time control of soft robots [CED19], provides an accurate and faithful simulation but with a limited simulation speed. Finally, the quality of the simulation depends on the choice of model, the calibraiton, and the consideration of the uncertainty of the simulated quantities.

The soft-robot simulations can be performed using various methods, each of which relies on a distinct modelling approach. In fact, simulation corresponds to the resolution of a model over time. Analytical models, such as the Rigid Body Model [KP13b] or Constant Curvature Model [WJ10], offer very fast solutions but rely on strong assumptions as explained previously. Heuristic models, like the Mass Spring

Models [LSH07] which represent the robot as a set of masses connected by networks of springs, also provide fast solutions, but the model parameters do not correspond directly to the robot's mechanical properties and thus require specific optimisation methods to be found. The last method involves simulating the behavior of soft robots using continuum mechanics equations. There are no general analytical solutions for these equations except for simplified and/or unrealistic cases [GC14; KP13a; GJC09; Hes+13; Mar+14; Fal+15]. Due to the lack of analytical solutions, numerical methods have been developed. FEM can capture complex mechanical behaviors (nonlinearities, heterogeneities, different actuators). It involves spatial discretization of the robot geometry, called a mesh, and the use of numerical solvers to solve the equations of deformation. These methods are slower than analytical solutions but allow the simulation of soft robots with fewer assumptions. Other methods exist, such as the Material Point Method (MPM) [Zha+15], which is a free-mesh method in which a continuum body is described by several small Lagrangian elements. This method uses a background mesh to calculate terms, such as the deformation gradient, without encountering the disadvantages of mesh-based methods (e.g., remeshing algorithms). However, it is more computationally expensive than FEM because the grid must be reset at the end of each MPM calculation step.



Framework	Modeling approach	Physics model complexity	Installation	User interface	URDF[*]	Rigid robots[*]	Continuum robots[*]	Rigid-soft hybrid robots[*]	Open source
Gazebo	rigid-body physics	low	Install ROS & compile	GUI, ROS (C++ or Python)	✓	✓	✗	✗	✓
MuJoCo	rigid-body physics	low	Executable	C++, Python	✓	✓	✗	✗	✗
PyBullet/Bullet	rigid-body physics	low	Python pip	C++, Python	✓	✓	✗	✗	✓
Webots	rigid-body physics	low	Executable	C, C++, Java, Python, MATLAB	✗	✓	✗	✗	✓
Elastica	Cosserat rods	medium-high	Python pip	C++, Python	✗	✗	✓	✗	✓
Huang et al. (2020)	Cosserat rods	medium-high	Compile source	C++	✗	✗	✓	✗	✗
ChainQueen	Particle-grid hybrid	high	Compile source	Python, Taichi	✗	✗	✓	✗	✓
SOFA	FEA	high	Executable	GUI, C++, Python	✗	✓	✓	✓	✓
PyBullet + SoMo	rigid-body physics	low	Python pip	Python	✓	✓	✓	✓	✓

[*]indicates native support

Fig. 2.2: Overview of popular robot simulation frameworks, table from [Gra+21].

Several simulators are available for modelling and controlling rigid robots, such as Gazebo and NVIDIA ISAAC. The options are limited to simulate soft robots [JH21]. Commercial software such as COMSOL, ANSYS, and Abaqus can model soft actuators. However, these software are slow and are more useful for simulating more general physical phenomena, and do not natively include the concepts we need for robotics, such as actuators, effectors, sensors, state spaces and execution of control laws [JH21]. Many physical simulators using an RL Gym interface [Bro+16], such as Mujoco [TET12], PyBullet [CB21], PlasticineLab [Hua+21b] and SoftGym [Lin+20], have been used to simulate interactions with deformable objects [MJD18;

Wu+19; Sei+20; Tag+20] but not soft robots. ChainQueen [Hu+19b] and Diff-Taichi [Hu+20] are differentiable simulators for simulating soft robots, but they are limited in their capabilities (heuristic approaches, limited actuators, structures, and constitutive laws). Elastica [Nau+20] is a simulator based on dynamics Cosserat model used to simulate slender robots. Finally, the Simulation Open Framework Architecture (SOFA) software [Fa12] is a FEM simulator co-developed by several INRIA teams, including the DEFROST team in which I conducted this PhD. It allows the consideration of multiphysics phenomena, collisions, and tools specifically created for soft robotics, such as soft actuators and inverse models [Tea19]. These tools enable the modelling, simulation, and control of soft robots [Coe+17]. Figure 2.2, from [Gra+21], gives an overview of these different simulators.

2.1.3 FEM simulation

In this study, we exclusively used FEM simulations of the soft robots. In Chapter 3, we will discuss the advantages and limitations of this choice. In the next section, we present the continuum mechanics equations employed in FEM simulations. These equations serve as the foundation for the condensed FEM model presented in Chapter 5. In addition, we introduce the notations used in this manuscript for the FEM-based models.

Usefull concept: convert the mechanical equations obtained from the FEM to a system $Ax = b$ at each simulation step, mechanical matrices (A, K, D), definition of the constraints as Lagrange multipliers, characteristic constraint distances δ_i, free configuration of the robot x_{free}.

FEM simulation without actuation and contact

The behavior of soft robots can be mathematically described using the mechanical equations of continuous media. FEM simulation software such as SOFA can be used to obtain an approximate solution to these equations. In the FEM simulations, the state of the soft robot is described as a discretized mesh, represented by a finite number of 3D points. When there is no actuation or contact, the dynamic equations are written as

$$M\ddot{q} = f_{int}(q, \dot{q}) + f_{ext} \tag{2.1}$$

where M is the mass matrix of the system, $\ddot{q}$, $\dot{q}$, q are the acceleration, velocity, and position of the nodes of the FEM mesh, $f_{int}(q, \dot{q})$ are the internal forces depending on the deformation model of the system (elastic, beam, ...), and f_{ext} are the external forces such as the gravitational force.

The dynamic equations are discretized in time using an integration scheme. An implicit integration scheme, such as the one employed in SOFA, ensures unconditional stability, but necessitates matrix inversion at each time step. Another important reason for choosing this scheme is that it is adapted to the 'time-stepping' used in non-regular mechanics to deal with speed jumps on contact. Let $h = t_{i+1} - t_i$ be the time step, $(q_i, \dot{q}_i)$ be the current state, and $d\dot{q} = \dot{q}_{i+1} - \dot{q}_i = h\ddot{q}_{i+1}$. It is possible to write the discretized version of equation 2.1 as

$$\begin{aligned} M d\dot{q} &= h f_{int}(q_{i+1}, \dot{q}_{i+1}) + h f_{ext} \\ q_{i+1} &= q_i + h\dot{q}_{i+1} \end{aligned} \tag{2.2}$$

Knowledge of the position and velocity at the end of the time step is necessary to obtain the internal forces. To overcome this problem, a first-order Taylor series expansion is used to linearize the internal forces:

$$f_{int}(q_{i+1}, \dot{q}_{i+1}) = f_{int}(q_i, \dot{q}_i) + \frac{\partial f_{int}}{\partial q} dq + \frac{\partial f_{int}}{\partial \dot{q}} d\dot{q} \tag{2.3}$$

We denote $\frac{\partial f_{int}}{\partial q} = K$ the stiffness matrix and $\frac{\partial f_{int}}{\partial \dot{q}} = D$ the damping matrix. In SOFA, Rayleigh damping $D = \alpha M + \beta K$ is used with α the Rayleigh mass and β the Rayleigh stiffness. By writing $dq = q_{i+1} - q_i = h\dot{q}_{i+1} = h(\dot{q}_i + d\dot{q})$, we obtain the system:

$$\underbrace{(M - hD - h^2 K)}_{A} \underbrace{d\dot{q}}_{x} = \underbrace{h f_{int}(q_i, \dot{q}_i) + h^2 K \dot{q}_i + h f_{ext}}_{b} \tag{2.4}$$

We obtain a sparse matrix system $Ax = b$. In mechanics, impedance is the quantity that multiplied by a velocity gives a force. Therefore, the matrix A can be interpreted as an integrated impedance, in the sense that the choice of the implicit integration scheme and the time step have an influence on the value of this matrix. Then, it is possible to determine the value of $d\dot{q}$ by solving this system. The values q_i and $\dot{q}_i$ are updated using the Euler scheme $\dot{q}_{i+1} = \dot{q}_i + h d\dot{q}$ and $q_{i+1} = q_i + h\dot{q}_{i+1}$. K, D

and M are symmetrical, positive-definite, and sparse; therefore, A is invertible. This formulation allows also the simulation of rigid robots [Tal+15].

FEM simulation with actuation and contact

We consider a force control of the robot. The actuation and contact constraints are represented by the Lagrange multipliers λ_a and λ_c. λ_a is set by the user or by the command. These constraints are integrated into 2.4 according to

$$Ax = b + hH_a^T\lambda_a + hH_c^T\lambda_c \tag{2.5}$$

H_a^T and H_c^T can be seen as the Jacobian matrix[2] of the constraint and depend on the type of constraint (cable, cavity, contact, etc.). More precisely, this quantity comprises the directions of the forces applied to the nodes involved in the constraint. It is a matrix of size $n \times n_i$ where n is the number of DoFs in the FEM mesh and n_i for $i \in \{a, c\}$ is the number of DoFs for the constraint. For the actuation, n_i corresponds to the number of actuation variables. λ_a and λ_c are vectors corresponding to the constraint variables. For example, in cable actuation, λ_a corresponds to the force applied to the cable. In pneumatic actuation, λ_a corresponds to the pressure inside the cavity. For each type of constraint, we define the characteristic distance δ_i for $i \in \{a, c\}$. This quantity is the length of the cables for cable actuation, expansion of the cavities for pneumatic actuation, and so on. In the general case, δ_c is the signed distance between two objects in the contact zone. This distance is positive or zero to avoid interpenetrations.

The constraints are solved in two steps:

1. Free motion: First, we find the free configuration x_{free} when there is no actuation or contact $\lambda_a = \lambda_c = 0$, which is obtained by solving $Ax_{free} = b$.

2. Correction: Next, we correct this value by incorporating the effect of the constraints. The final solution is given by $x = x_{free} + hH_a^T\lambda_a + hH_c^T\lambda_c$.

The formulation of the constraints in the form of Lagrange multipliers is used for both direct and inverse simulations. In SOFA, contact points and surfaces are detected at each time step using collision detection algorithms, and the responses to the

[2]In robotics, the Jacobian matrix links the speeds of the actuators to the linear/angular speed of the end-effector. Here we are using Jacobian matrices in the mathematical sense of the term, i.e. the first derivative of a vector function.

constraints are calculated using Signorini's law and Coulomb's law. A comprehensive description of the collision pipeline can be found in [Fa12].

2.2 Control in soft robotics

Due to their compliance, it is challenging to control soft robots as their behavior is highly nonlinear and they are subject to large deformations. When the task involves to follow identified target positions and velocities with the robot effector, inverse modelling based on optimisation methods can be used to control the robot [CED19; Coe19]. However, these targets are difficult to define when the robot must perform tasks other than positioning, such as grasping and manipulating an object, or walking efficiently in a given direction. In this manuscript, we make the distinction between what we call high-level control and low-level control: low-level control corresponds to the following of a trajectory by a robot, while high-level control corresponds to the generation of this trajectory to solve a given task. Although there are different definitions of high-level tasks (for example, tasks with numerous different motion to perform and tools to use such as cooking), we restrict the focus to tasks with sequential configurations and contacts. If optimisation approaches are efficient in solving low-level tasks, they cannot solve high-level tasks alone and must be coupled with other algorithms [Bra+07] such as the Rapidely-exploring Random Tree (RRT) path planning algorithm [MGK22; Haw+19].

Another way to solve these high-level tasks is to use learning algorithms such as Reinforcement Learning (RL) algorithms. RL algorithms are used in many domains such as video games [Nic+18] or virtual environments [Mni+15; Haf+18]. In robotics, advances in artificial intelligence have enabled the creation of autonomous robots that can perform more complex tasks [JP13], such as pick-and-place tasks [PG15; Lev+16; Agr+16] or manipulating Rubik's cubes with a robotic hand [Ope+19b; Ope+19a]. Although the application of RL algorithms to soft robotics presents unique challenges [Bha+19a], they have been used to solve manipulation [Hom+18] and locomotion tasks [Wan+16]. Other learning algorithms, such as supervised learning algorithms and variations of model-based and model-free RL algorithms, have also been used for control in soft robotics [WLK21]. However, learning methods are often criticized for being less interpretable and less sample efficient than optimisation-based control methods. As explained in 2.2.2, various methods can be used to address these problems [Wil+22]. In the following sections, we present two methods for controlling a soft robot (sequential decision problem and optimisation-based control) and their link with simulation.

2.2.1 Simulation and control in soft robotics

The control methods described above can be combined with a robot simulation. In the case of optimisation-based control, the simulation can be used to compute robot-specific mechanical quantities required for optimisation. This is the case in the FEM simulation of soft robots, where different matrices, such as the integrated impedance A, are used for the control [CED19]. Learning-based approaches can also benefit from simulations. These approaches use a large amount of training data to learn the optimal strategy. However, these data are not always easy to obtain using a physical prototype of the robot. There are many advantages to simulation: parallelization and computation time faster than real-time, no risk of damaging a physical prototype in the trial-and-error process, and training in critical tasks. However, the trade-off between the accuracy and speed of the simulation is important. When an accurate simulator is used, the complexity of the simulation (many contacts or constraints) leads to very high computation times, which limits the use of machine-learning algorithms. When simulation is used for control, the strategy is learned [Cho+21] or calculated in simulation before being transferred to a physical robot.

2.2.2 Sequential decision problem and control in soft robotics

Control of a soft robot can be achieved using different methods. The first method considered in this work is to control the robot using machine learning. The objective of this section is to mathematically formalize sequential decision problems and determine how RL and planning algorithms can solve them. These concepts will be relevant in Chapter 3 when we discuss the SofaGym API, which couples the FEM simulation and sequential decision problem, as well as in Chapter 4 where we will use the concepts of observation space, trajectory, and value function. In addition, we introduce the notation used in this study for machine learning topics.

Usefull concept: MDP, state space S and action space A, reward function $r(s, a)$, options, strategy, value function V and Q-function Q, Reinforcement Learning and Planning algorithms.

Definition of a sequential decision problem

Solving a sequential decision problem consists of determining a sequence of decisions to achieve a certain goal. An important class of such problems are Markov Decision Problems (MDP). A MDP is defined by a state space S with $s_t \in S$, an action space A with $a_t \in A$, a transition function $s_{t+1} \sim \mathbb{P}(s_t, a_t)$, an immediate reward function $r_t \sim r(s_t, a_t)$ and an objective function J to be optimized. An example of a MDP in the context of soft robotics is shown in Figure 2.3.

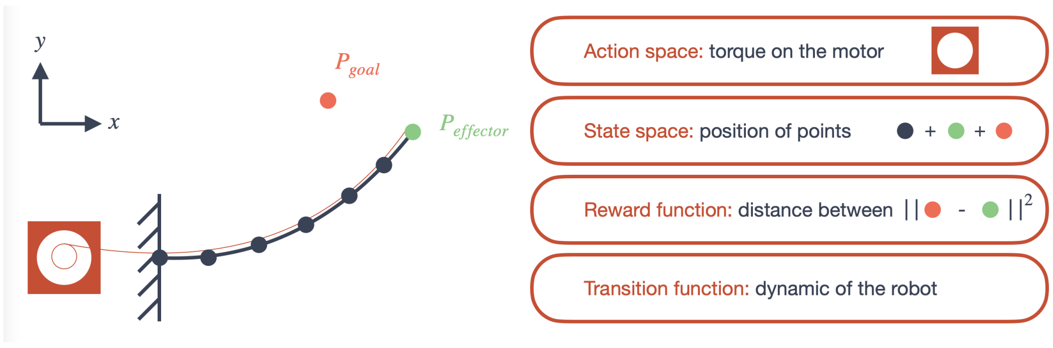

y
x
P_{goal}
$P_{effector}$
Action space: torque on the motor
State space: position of points ● + ● + ●
Reward function: distance between || ● - ● ||^2
Transition function: dynamic of the robot

Fig. 2.3: Illustration of a simple MDP for a trunk-like robot. The action space is composed of the possible values of torque applied to the motor (red square). The state space includes the positions of the different points of the robot (gray dot), the effector (green dot), and the goal to reach (red dot). The reward is the Euclidean distance between the end effector and the target in the 2D plane (x, y). The transition function is given by the dynamic of the soft robot.

It is common to work with both continuous and discrete state and action spaces. In robotics, the action space can consist of various values imposed on actuators, such as servomotors, linear actuators, cables or pneumatic cavities. The state of the robot can be described using sensor information such as visual information from a camera. However, not all relevant state information may be accessible, so the term "observation" is often used to refer to the state information that is available for solving a given task.

Solving an MDP, and thus the corresponding task, involves finding a strategy $\pi(a|s)$ that represents the probability of performing action a in state s. A widely used objective function represents the expected cumulative reward:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}[\Sigma_{t=0}^{T} \gamma^t r(s_t, a_t)] \tag{2.6}$$

where $\mathbb{E}$ is the mathematical expectation, $\gamma \in [0, 1)$ is a discount factor [Put94], $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...)$ is a trajectory sampled from π and T is the time horizon. It is assumed that time is discretized $t \in \mathbb{N}$ and that the effect of action a_t is finished by time $t + 1$. Under these conditions, MDP theory states that there is an optimal strategy π^* that maximizes J.

Finally, an option [SPS99] is defined as a sequence of primitive actions, which are one time-step-long actions. Options consist of three elements: a strategy π, a termination condition $\beta(s)$, and a starting set $I \subseteq S$. Option $< I, \pi, \beta >$ is considered available in state s if $s \in I$. If an option is selected, actions are chosen according to strategy π until the termination condition is reached with probability β. Once an option is complete, the agent can choose another option. An MDP with a finite set of options is called a semi-MDP (SMDP).

Solving a MDP using RL algorithms

If we know the elements of the MDP, finding the optimal strategy is an optimisation problem. In the case of RL, the sets S and A as well as the cost function J are known, but not the transition or reward functions. The agent interacts with the environment to obtain information regarding these two functions. It learns the consequences of its actions in its current state (the next state and immediate reward) through a trial-and-error approach. The interaction between the environment and the agent is illustrated in Figure 2.4.

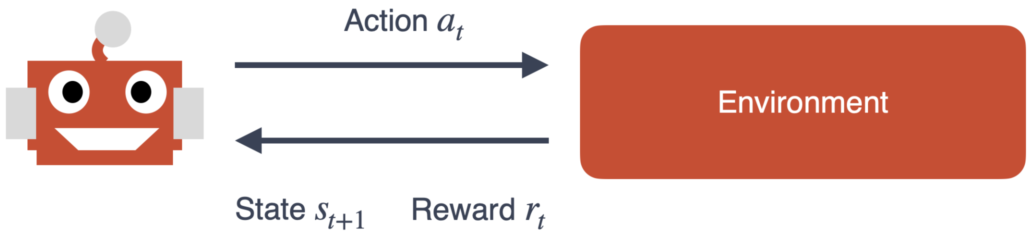


Fig. 2.4: Illustration of RL algorithm function. An agent interacts with an environment via an action, and, given a current state, it receives information about the next state (transition function) and the immediate reward (reward function).

Different methods exist to solve this optimisation problem. Commonly, these methods are based on the value function V defined as:

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi}[\Sigma_{t=0}^{T} \gamma^t r(s_t, a_t) | s_0 = s, a_t = \pi(s_t)] \tag{2.7}$$

This function is similar to the cost function defined in the equation 2.6 but it depends on a state s and defines the expected outcome of starting from a state $s_0 = s$ and following the strategy π. It can be considered as a measure of interest in a given state, because it quantifies the expected reward from that state. From MDP theory, a strategy is optimal if it maximizes V for any initial state. Instead of working with this function, it is possible to work with the function Q defined by

$$Q_\pi(s,a) = \mathbb{E}_{\tau\sim\pi}[\Sigma_{t=0}^{T}\gamma^t r(s_t,a_t)|s_0 = s, a_0 = a, a_t = \pi(s_t)] \tag{2.8}$$

where we recognize the reduced sum of rewards by starting from the state $s_0 = s$ and choosing action $a_0 = a$. For an optimal strategy, the optimal action choice corresponds to choosing action according to $Q_{\pi^*}(s,.)$ that maximizes the expected reward. Calculating $Q_{\pi^*} = Q^*$ is sufficient to determine how to act optimally. It is possible to calculate π^* using value iteration algorithms [Bel57] and policy iteration algorithms [How60].

In general, the value of the Q-function is unknown and methods such as Q-learning are used to approximate it. Within the Deep Learning paradigm, the Q-function is represented by a neural network (NN) [Tes95; RN94; Lin92; Gul90; Tha94]. This method is the foundation of many algorithms such as the Deep Q-Network (DQN) [Mni+13] and Double DQN [HGS15]. Other methods represent the strategy using a learned function parameterized by θ, for example, using a NN. This is the case for actor-critic approaches such as the Asynchronous Actor-Critic algorithm [Mni+16], Soft Actor Critic algorithm (SAC) [Haa+18], Proximal Policy optimisation algorithm (PPO) [Sch+17] and Deep Deterministic Policy Gradient algorithm (DDPG) [Lil+15].

Improved sample efficiency and generalization in RL algorithms

There are two main challenges associated with RL algorithms: generalization and sample efficiency. Generalization refers to the ability to continue to perform optimally in states not encountered during the learning phase. Sample efficiency refers to finding the optimal solution with the least amount of interaction with the environment. Improving the sample efficiency and generalization of RL algorithms is possible, and there are several approaches to achieve this. One approach is Model-Based RL, in which a model of the environment is learned simultaneously with the control strategy/Q-function. Transitions between states $(s_t, a_t) \rightarrow s_{t+1}$ are sampled, for example, using a random controller [FL16], and used to learn the transition function $s_{t+1} \sim \mathbb{P}(s_t, a_t)$. Another way of dealing with sample efficiency, generalization, and interpretability is to add prior knowledge into the network [Wil+22]. One way to do this is to add a penalty to the cost function related to the physical behavior of the agent [Kar+17; Ebe+21]. This approach has been applied to inverse modelling tasks [Kah+20], solving differential equations [YZK20] and general control tasks [Lof+15; Hu+19a; DI19; ELS21]. Another way is to initialize the models with the prior knowledge [Bha+19b; Jia+21]. Initialization is a key aspect of Transfer Learning and Imitation Learning [Hua+21a; Tob+17; Bou+18]. The idea behind Transfer Learning is to learn how to solve one task and to use this learning to solve another task. This is the case when the solution to a task is learned in the simulation before being transferred to a physical prototype. Imitation Learning consists of learning behavior from an expert's trajectory. A third way to incorporate prior knowledge is to group similar states. These methods can be used to regularize RL algorithms [Akr+18], facilitate state exploration [Man+04], or create states from large spaces such as images [Yar+21; DJA22].

Hierarchical Reinforcement Learning (HRL) [Pat+21] can also be used to facilitate the learning process. The objective of HRL is to split a task into different subtasks that are easier to solve. The problem is usually decomposed into two parts [Hen17]: a high-level strategy chooses which subtask to solve, and a low-level strategy solves it. This method has been applied to the theory of continuous control [FDA17; Lev+19] and robotics in manipulation tasks [Fox+17; Gup+19; VAP20]. HRLs are formalized mathematically in the SMDP theory presented in Section 2.2.2 and there are many methods to solve them: the Feudal Hierarchy approach [DH92; Kul+16; SPS99] where the high-level strategy distributes tasks to workers via sub-goals; Hierarchy of Abstract Machines [PR97] where finite state machines are used to solve a given task; or multi-agent approaches such as Multi-Agent Hierarchical Reinforcement Learning (MAHRL) [AD19; Low+17; Lev+19] where several agents solve a task by distributing subtasks to each other. The use of different subtasks that are easier to solve than a high-level task speeds up learning [LS11; Rus+16], in particular by sharing information between all the subtasks (states, action, Q-function, etc).

Solving a MDP using planning algorithms

Planning can be solved as a sequential decision problem. The objective of planning algorithms is to create a plan to achieve a specific goal. Planning algorithms operate "offline," meaning that the plan is formulated before any action is taken in the environment. This requires having access to the plan before interacting with the environment, which necessitates a model of the environment to predict the consequences of the chosen actions. The mathematical formulation of this concept is an MDP, where the state set S, action set A, transition function, reward function, and objective function are known.

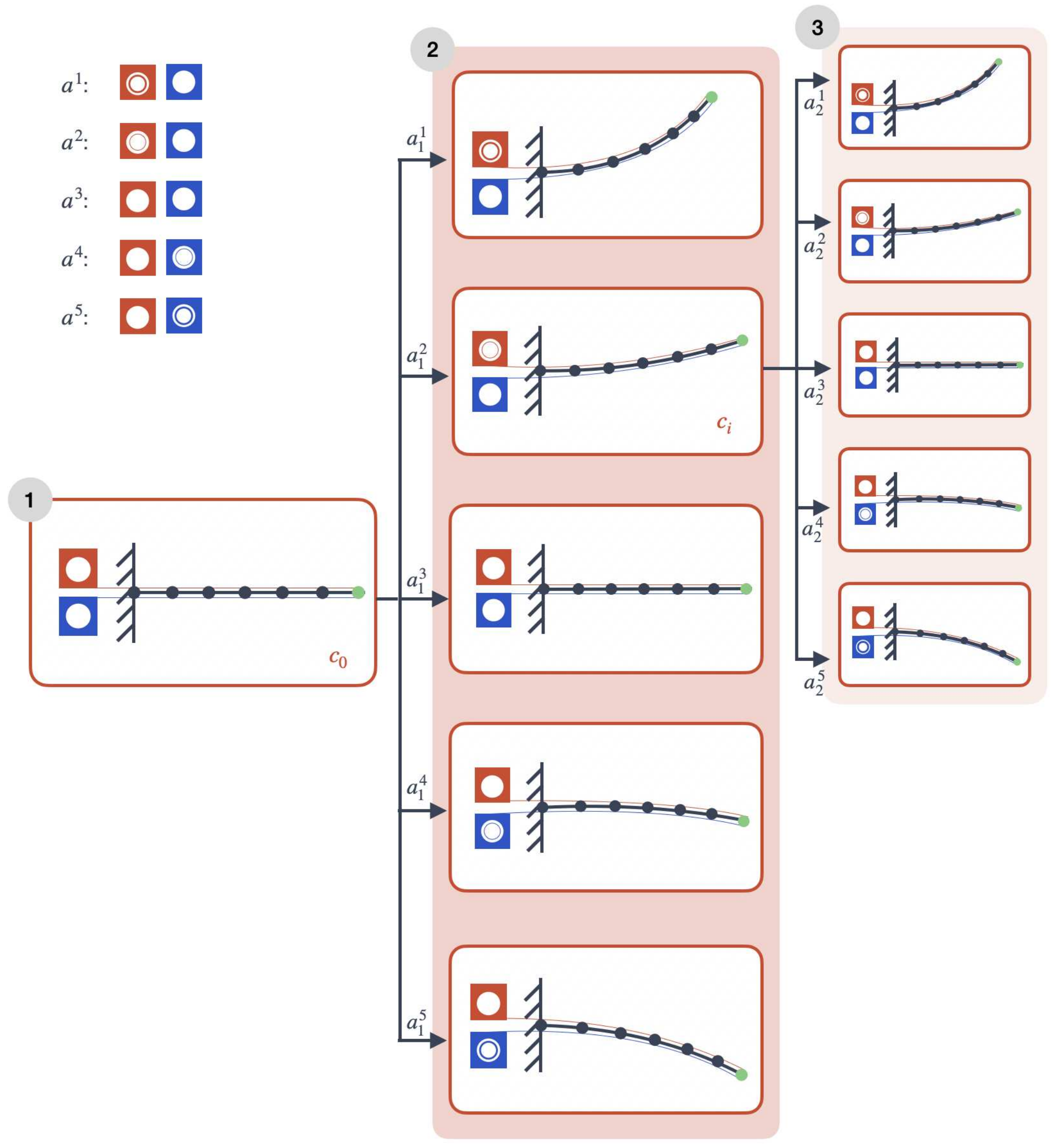


Fig. 2.5: A visual representation of a tree search algorithm for a trunk-like robot (dark gray in the image) with five actions. These five actions represent five different motor configurations (red and blue squares in the image). Starting from an initial configuration c_0 (1), five actions are tested (2), which yields five new configurations of the robot. A leaf c_i is selected for expansion (3). The best path in the tree leads to the lowest reward.

When such a model is available, sampling-based optimisation methods can be used to determine a control strategy. When action space is discrete, Tree-search algorithms like Monte Carlo Tree Search (MCTS) methods are commonly used. These methods are illustrated in Figure 2.5. They construct a tree of possibilities defined by the following rules:

- The root of the tree c_0 is the initial configuration of the environment.

- Each node of the tree c_i is the configuration obtained by starting from the previous configuration $c_{p(i)}$ and playing action a^i with $p(i)$ the parent of node i in the tree.

Once these elements have been defined, a tree can be developed using the following steps: selecting the most suitable leaf based on predetermined criteria, simulating one or more actions starting from the leaf, adding new leaves to the tree, calculating the reward from the new leaves, and backpropagating the expected reward to the rest of the tree. The tree is traversed until it reaches the terminal state.

There are many tree-search algorithms [LaV06]. These include the Upper Confidence Bounds applied to trees (UCT) [KS06] from the Multi-Armed Bandit community, Open-Loop Optimistic Planning (OLOP) [BM10] and Optimistic Planning for Deterministic systems (OPD) [HM08]. These algorithms aim to extend a tree search to find the sequence of actions that solves a given task, but they differ in how the leaf to be extended is chosen. For example, OPD is based on the principle of Optimism in the Face of Uncertainty [Mun14]. In this algorithm, the lower and upper bounds of the value function are calculated for each state in the tree using the reward and the Bellman operator (see Section 2.2.3) for backpropagation [Leu20]. Based on these bounds, the sampling rule follows the action that maximizes the upper bound (optimistic sampling rule), and the recommendation rule selects the action that leads to the state with the highest lower bound (conservative recommendation rule). Other methods exist for continuous action domains, but they are outside the scope of this discussion.

2.2.3 Optimisation problem and control in soft robotics

The second method for controlling a robot is to use mechanical modelling and optimisation methods. The objective of this section is to give the mathematical expression of optimal and inverse control and explore how to apply it in robotics. It will be useful to understand the following chapters. In Chapter 3, we compare the learning-based and optimisation-based methods. In Chapter 4, we combine learning methods and optimisation methods for both proxy and configuration spaces graph topics. Finally, in Chapter 5, the control is realized by writing a new optimisation problem based on the following development. We also introduce notations that are used for optimisation-based control.

Usefull concept: Optimal control, Bellman equation, forward and backward steps, effectors, projection in the constraint space, quadratic programming for control.

Optimal control and trajectory optimisation

We consider a multistep control problem, that is, the problem of finding a robot trajectory that minimizes a given cost function. Mathematically, multistep control can be formalized as an optimal control problem. Using notations from the literature [GD19; Bud+18; JMC22; SP08], an optimal control problem can be written in its simplest form as

$$\begin{aligned} &\min_{x,u} \int_0^T l(t, x(t), u(t))dt + h(x(T)) \\ &\dot{x}(t) = f(t, x(t), u(t)) \\ &x(0) = \bar{x}_0 \end{aligned} \tag{2.9}$$

with t the time, $T > 0$ the time horizon, $x(t)$ the state of the system at time t, $u(t)$ the actuation of the system at time t, $\bar{x}_0$ the initial state of the system, f the continuous dynamics of the system $l(t, x(t), u(t))$ the cost function at time t (running cost) depending on the state and the actuation and $h(x(T))$ the terminal cost function depending only on the state at time T. Constraints can be added to the validity of the pair $(x(t), u(t))$. It should be noted that $u = \lambda_a$ in the equation 2.5. Here, we use notation u as it is commonly used in the literature on optimal control.

As it is generally done in the numerical optimal control literature [GD19], problem 2.9 is discretized according to

$$\begin{aligned} \min_{x,u} J(x,u) &= \sum_{k=0}^{N-1} l_k(x_k, u_k) + h(x_T) \\ x_{k+1} &= f_t(x_k, u_k) \\ x_0 &= \bar{x}_0 \end{aligned} \tag{2.10}$$

with $T = N\Delta t$, Δt the temporal time step, $x_k = x(k\Delta t)$ and $u_k = u(k\Delta t)$ the discretization of the state and actuation, l_k the discretization of $\int_{k\Delta t}^{(k+1)\Delta t} l(s, x(s), u(s))ds$, and f_k the temporal discretization of the dynamics f using an implicit or explicit temporal discretization scheme.

Differential Dynamic Programming (DDP) [MAY73] is a popular optimal-control algorithm that can efficiently solve problem 2.10. The solution involves the Bellman equation, which provides the recursive formula:

$$V_k(x) = \underset{u}{min}\ l_k(x,u) + V_{k+1}(f(x,u)) \tag{2.11}$$

where V_k is called the value function at time k and $V_N(x) = h(x)$. Like the equation 2.7 for sequential decision problems, the value function recursively represents the cost calculated at iteration k for state x.

To solve this equation, the DDP calculates a Newton step according to control variable u for a certain value $x + \delta x$ of the previous state x_k. This is the backward step that allows the generation of a new sequence u from a nominal trajectory. The Hamiltonian function $Q(x,u) = l(x,u) + V_{t+1}(f(x,u))$ is used to calculate the value function. It is possible to write a quadratic approximation of Q, according to $Q(x+\delta x, u+\delta u) - Q(x,u)$, as a function of the data of the problem, that is, l, f and V_{k+1} and of their first and second derivatives. In the case of problem 2.11, it is possible to minimize this quantity using the Schur complement Lemma and to find the value of the command u that corresponds to this minimization. Once the backward step has been performed, that is, once the u sequence has been found, it is possible to perform a forward step. The forward step allows to calculate and evaluate a new nominal trajectory from a given control sequence. The forward step is obtained using a nonlinear rollout. The actuation and state values are updated using a backtracking line search procedure. Thus, the combination of backward and forward steps allows to perform multistep control by first finding a command sequence u to improve a nominal trajectory, and then using this command sequence u to find a new nominal trajectory. Several successive backward and forward steps may be necessary to obtain the solution. An illustration of this process is presented in Figure 2.6.

The present method considers a problem in which the only constraint is given by the explicit dynamics of the system evolution, but it is not the only formulation. For instance, it is not necessary to solve the dynamics at each time step because the problem can be solved by approximating the dynamics, leading to the use of implicit dynamics in problem formulations [JMC22]. Additionally, equality or inequality

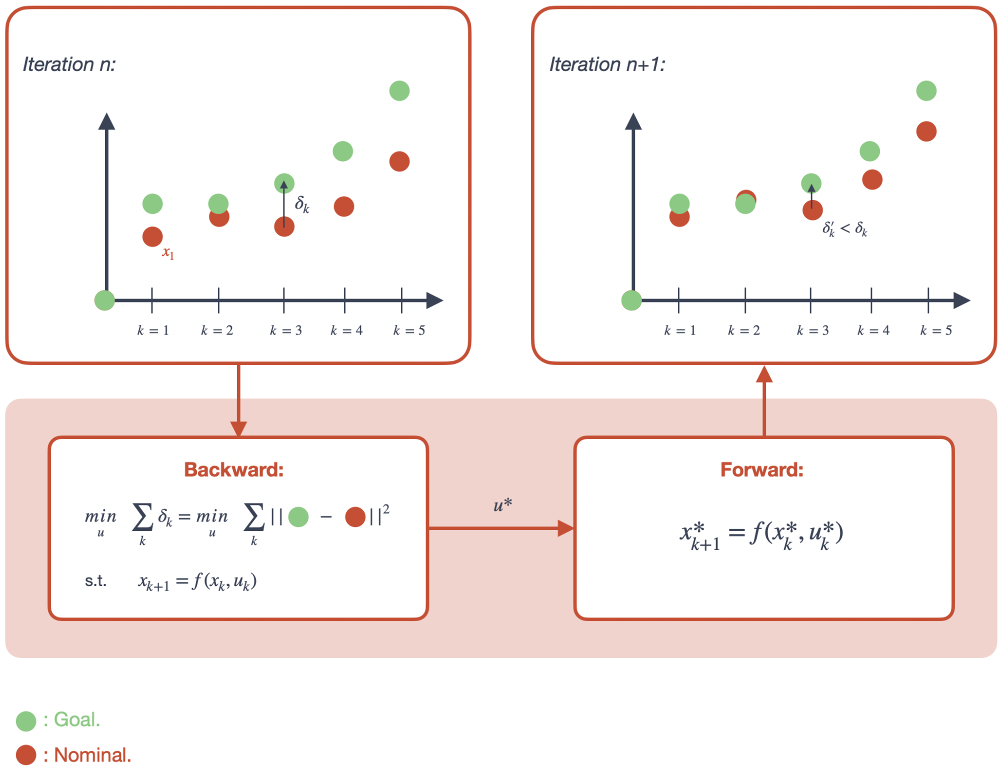


Fig. 2.6: Illustration of optimal control problem resolution. The goal is to minimize the distance between the nominal trajectory of points x_k (red points) and an objective sequence (green points). The backward step computes an action sequence u^* that minimizes the sum of the point-to-point distances δ_k between the trajectories. The forward step then computes a new nominal trajectory defined by states x^*.

constraints can be incorporated into the problem formulation. [JMC22]. With inequality constraint and implicit dynamic, the problem 2.11 becomes

$$
\begin{aligned}
V_k(x) &= \min_u l_k(x,u) + V_{k+1}(y) \\
f(x,u,y) &= 0 \\
lb &\leq C_1 x + C_2 u \leq ub
\end{aligned}
\tag{2.12}
$$

where lb and ub are lower and upper bound, C_1 and C_2 some matrices involved in the inequality constraints. This problem is solved using the same method, but the solution to the optimisation problem is found using sequential quadratic programming (SQP) [NW06].

The use of multistep control in soft robotics is currently limited to simplified models of soft robots, such as piecewise constant curvature [SK22], model order reduction formulations [TLP20], or spectral submanifold[3] only valid around a specific equilibrium point [Alo+22].

Quadratic programming for control in soft robotics

In soft robotics, a method for one-time step control has been proposed by [Coe19]. This method is based on inverse modelling of the robot and a condensed FEM description of its behavior. This method has already been implemented in SOFA [CED17]. We present this method in cases where contacts have to be considered.

The movement of a soft robot is controlled by specific points called the effectors. A common method to perform a task is to control the positions of these specific points. The idea is to minimize the distance between the effectors and a goal, which can be described by a characteristic distance $\delta_e = p - p_{goal}$ where p is the position of the effectors. We can define the Jacobian matrix of the effectors. For an effector with three degrees of freedom in position, we have:

$$H_e^T = \begin{bmatrix} 0 & \dots & 0 & I & 0 & \dots & 0 \end{bmatrix}^T \tag{2.13}$$

where the identity matrix is at the control-point position indices. The generalization to several control points is straightforward.

As with the forward modelling described in 2.5, the solution of the inverse problem is divided into two steps. The first step involves determining the free configuration of the robot, that is when all constraints $\lambda = 0$. Once the system is solved for x^{free}, it is possible to evaluate δ_i^{free} for $i \in \{a, e, c\}$ which corresponds to the violation of the constraints in the free configuration of the robot. The idea is to correct this value using actuation by making a linear approximation of the displacement of the nodes based on FEM interpolation. This is achieved using the projection of the integrated admittance matrix A^{-1} (Eq. 2.4) in the constraint space [Coe19]:

$$\begin{aligned} \delta_e &= W_{ea}\lambda_a + W_{ec}\lambda_c + \delta_e^{free} \\ \delta_a &= W_{aa}\lambda_a + W_{ac}\lambda_c + \delta_a^{free} \\ \delta_c &= W_{ca}\lambda_a + W_{cc}\lambda_c + \delta_c^{free} \end{aligned} \tag{2.14}$$

[3]Nonlinear extension of a spectral subspace of a linear dynamical system under the addition of nonlinearities

with $W_{ij} = H_i A^{-1}(x) H_j^T$ for $i, j \in \{e, a, c\}$ the projection of A in the constraint space. The matrices W_{ij} for $i, j \in \{e, a, c\}$ represent the mechanical coupling between the effectors e, the actuators a and the contacts c. This set of equations represents a condensed model of the FEM model (unlike the reduced one), in the sense that the entire dynamics is projected into the constraint space which dimension is usually lower.

It should be noted that this scheme is already used in the forward simulation. The difference is that the actuation values are selected by the user and directly applied to the actuators. In inverse modelling, the values of the actuation and the contacts are not known, and the following optimisation problem must be solved to find them:

$$\begin{aligned} \underset{\lambda_a, \lambda_c}{min} \quad & ||\delta_e||^2 \\ s.t.\ \delta_{min} \leq \delta_a \leq \delta_{max} \quad & \text{(Actuators course constraint)} \\ \lambda_{min} \leq \lambda_a \leq \lambda_{max} \quad & \text{(Actuation effort constraint)} \\ 0 \leq \lambda_c \perp \delta_c \geq 0 \quad & \text{(Contact constraint)} \end{aligned} \tag{2.15}$$

Without the contact conditions, this is a Quadratic Programming (QP) problem. When contact conditions are used, it is considered as a QPCC (Quadratic Problem with Complementarity Conditions). Dedicated solvers are available to solve this type of optimisation problem. This control method of soft robots is based on a condensed FEM model of the robot using the quantities W and δ^{free}. The optimisation problem minimizes the distance δ_e, while respecting the constraints on actuation and contacts. However, this formulation corresponds to one-time step control and does not take into account changes in the contact configuration (activation and deactivation of contacts).

2.2.4 Short conclusion on control in soft robotics

This section provides a brief overview of the most commonly used methods for control in soft robotics. We have seen that it is possible to achieve this control either by methods related to sequential decision problems, or by optimisation methods. Each of these methods has its advantages and disadvantages.

2.3 Design optimisation in soft robotics

Although design is not the main focus of this PhD work, we will explore in Chapter 5 how the FEM condensed model can be used to address parameter optimisation problems and calibration. This section provides a short overview of the various approaches used in soft robotics for design optimisation.

Usefull concept: Design variables, gradient based algorithms.

The use of soft materials in soft robots increases the complexity of the system due to the strong coupling between geometry, materials, and actuation. Even if soft robots were initially designed by hand, by replacing rigid parts with soft parts or by copying nature's behavior, efficient exploration of possible designs requires the use of tools and simulation [CW20]. The simulation allows to test different designs in a virtual environment, to limit the number of iterations on physical prototypes and to consider the characteristics of the problem (such as complex geometry and material properties). In addition, the differentiability and mechanical modelling of the simulation can be leveraged to efficiently explore the design space. However, the simulation must be sufficiently realistic to generate a design that behaves close to its physical counterpart. The exploration of designs is highly dependent on the description of the state of the soft robot, because it determines the parameter and the shapes that can be represented.

The mathematical formulation of design optimisation for robots involves several key points [CW20; RT15]. The first is the formulation of the design goal, which is usually achieved by decomposing a high-level task (catching an object, moving in a direction, etc.) into a sequence of movements that the robot should be able to perform (e.g., bending movements in a grasping task). Second, a mathematical representation of the design variables is established to create a design space to be explored. Finally, the designs are tested and evaluated using analytical and/or simulation tools. In this work, we use the FEM simulation and SOFA software. The simulation choice has a major impact on the computation time of the optimization process.

An optimisation method is used to solve the design problem expressed in terms of the design variables and objective function. The selection of the optimisation method depends on the characteristics of the system. We focus on two main families of solutions:

- Gradient-based algorithms. These methods find good research directions, but require a fully differentiable problem. They are sensitive to initial conditions and can easily fall into a local minimum.

- Black-box methods, like Bayesian algorithms and evolutionary algorithms. In particular, evolutionary algorithms have become popular for efficiently exploring design spaces. However, their convergence speed is low when the problems have high dimensions.

2.3.1 Gradient-based approaches

The design can be optimised using gradient methods. For continuous robots, these methods are applied to a predefined parameterisation [Ber+15; Ros+19]. For robots with more complex geometries, most approaches found in the literature rely on topology optimisation tools [Che+21; LCY22; Wan+20; Zha+17], which are widely used numerical methods for structural optimisation. Unlike other methods, density-based Topology optimisation does not require users to provide the design and parameter specifications. Instead, it optimizes material layers in a predefined domain. The advantage of Topology optimisation is that it can generate new designs without human bias. Although this method has been used to optimize compliant component designs [Sig97], cable-driven grippers [Che+18] and co-optimize design and control [Spi+19], further development is required to account for the nonlinearities and incorporate actuation into the framework.

2.3.2 Evolution-based approaches

Evolutionary approaches are used to explore the design of soft robots [PH22; MMD19]. The concept behind this approach is to use the principles of Darwinian evolution to develop a design that improves its efficiency in completing a particular task, that is to minimise an objective function [SG19]. The general idea is to define a population of individuals, each of which is described by a genotype that corresponds to the design parameters. Each individual is evaluated according to a cost function that depends on the genotype. At each iteration, new individuals are created through mutations (a gene is randomly modified into one or more individuals) and crossovers (two parent individuals form a new individual whose genotype is a combination of the genotypes of both parents). Only individuals that minimize the cost function are retained for a new iteration of the process.

In soft robotics, these evolutionary approaches have been mainly used with the mass-sping-based particle simulator VoxCad [PH22; Sui+19; Gu+19]. Although the designs are original, they are difficult to manufacture in practice. Other research works base their design on high-level parametrization, which ensures that a valid design is obtained, that is, one that bridges the gap between simulation and reality. However, the limitations of these methods are that they require significant human effort for parameterisation, and the exploration space is reduced. An example of this is shown in [SSW22], where an evolutionary approach is combined with RL to perform a co-optimisation control design for a locomotion task. The search space is limited to the position of the robot's legs on the main body and cavities for actuation.

2.3.3 Short conclusion on design optimisation in soft robotics

This section provides a brief review of the most widely used methods for optimising the design of soft robots. The methods proposed so far need to be extended to capture the complexity of the robots' design as well as the environment in which they are used.

2.4 State description for learning and planning

This section 2.4 presents two methods for learning or computing a compact description of a robot's behavior, which are relevant to the next part of the manuscript. This topic is central to Chapter 5, in which the goal is to learn a condensed mechanical description of the state. In the first part of Chapter 4, the objective is to use a proxy to learn a strategy before transferring it to the FEM model of the robot.

Usefull concept: State Representation learning, simplified model in humanoid robotics.

2.4.1 Learning a compact representation

State Representation Learning (SRL) algorithms [Les+18] are used to learn small-dimensional features that fully describe a system. These methods are widely used to build compact descriptions of robots when many sensors are available to obtain information. These small-dimensional descriptions are generally used in RL algorithms for control purposes. The objective is to keep only descriptive information useful to the task at hand. This allows the manipulation of small dimensional data, which leads to better performance in terms of computation compared to approaches where all information about the robot is used.

Mathematically, the same framework as RL is used, where an agent can interact with an environment through action a_t that changes the state of the agent from s_t to s_{t+1}. The agent does not have knowledge of its state, but only of an observation o_t belonging to a certain observation space. The goal of SRL is to learn a low-dimensional representation $\widetilde{s}_t$ of state s_t from previous observations $o_{1:t}$. Therefore, the goal is to determine a mapping ψ such that $\widetilde{s}_t = \psi(o_{1:t}) \approx s_t$.

In machine learning, particularly in robotics, various SRL methods have been implemented [Les+18]. In [GD18], Proper Orthogonal Decomposition is applied to reconstruct the state of a robot based on deformation modes. This method requires an offline learning phase and is coupled with additional control tools for real-time control of soft robots. The resulting reduced model is design-dependant and can not be explicitly linked to geometric design parameters. Additionally, the reduction is dependant of the mesh. In [Ber+20], a data-driven approach is used for learning a soft robot open loop controller on a specific task scenario. A differentiable inverse model of a soft robot's quasi-static physics linking actuation to effector positions is learned. Since this direct link is dependant of the given scenario, the learned reduced model is only usable for it. Other methods like the use of robotic prior knowledge in the cost function to constrain the state space [Les+17] or the introduction of relational biases directly into the data [Kim+21b; Alm+21] can also be used. These methods have their own advantages and disadvantages. They allow the creation of reduced models of robots and provide tools to learn or reconstruct the robots' state.

2.4.2 Humanoid Robotics

It is possible to use humanoid robotics as an example to develop new model reduction methods for planning and learning in soft robotics. A humanoid robot is a good example of a complex robotic system with many degrees of freedom and redundant actuation. Humanoid robotics showcases the application of various robot models for motion-planning tasks. Many works rely on reduced models such as the centroidal model to plan motion without computing the entire dynamics of the robot [OGL13; DVT14; Mas+17; MLH10]. The idea is to use these simplified models to generate an admissible robot trajectory at low computational cost. An example of the simplified models from [Bud+18] is shown in Figure 2.7.

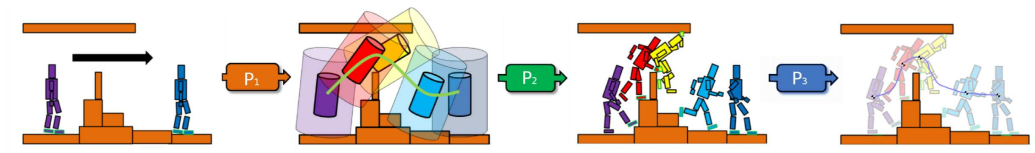


Fig. 2.7: Illustration of the use of simplified models in motion planning in humanoid robotics, from [Bud+18]. From left to right: defining an admissible trajectory with the robot's base, planning the contact sequence from the trajectory, and calculating the actual actuation required to follow the contact sequence using optimal control methods.

The purpose of this pipeline is to enable the humanoid robot to perform locomotion tasks. The search for the trajectory is not performed with the complete dynamics of the robot but rather with a reduced model that represents its root, where the limbs are attached. The trajectory is computed in the configuration space of the root (P1), and respects an equilibrium condition for the entire body. The path is then extended into a contact sequence (P2) that is feasible and that respects the root trajectory. Finally, an optimisation step (P3) is used to determine the actuation that allows the robot to follow this contact sequence. These models only consider a sub-part of the global trajectory generation problem. However, these models are linked with the complete model via the definition of admissibility constraints: the position of the root during the trajectory and the contacts between the robot and the ground must correspond to accessible positions.

2.5 Conclusion and positionning of this PhD study

In this first chapter, we have reviewed various tools that form the basis of the different works presented in this PhD thesis. We have seen that different descriptions of robots are used to study their behavior, control them, or design them. The objective of this PhD is to explore different representations of the state of a soft robot (modelling and parameterisation) in order to perform efficient control over several time steps. This will create the possibility of performing more complex tasks with soft robots in the futur.

At the beginning of this PhD thesis, there was no existing platform for effectively integrating RL algorithms with FEM simulations. As such, we explored the link between soft robotics, FEM modelling, and sequential decision algorithms to determine the potential and limitations of these approaches relative to traditional control methods. This is discussed in Chapter 3.

We are interested in two control solutions: learning-based and optimisation-based. Optimisation approaches can only be used efficiently if the robot is described in a convex task space. In addition, in soft robotics the optimisation-based methods are used for one-time step control. For RL algorithms, the observation must contain sufficient information to consider the deformations of the robot and the contacts between the robot and its environment. However, this observation cannot be large for an efficient learning step due to difficulties in parameterisation and computational efficiency. In Chapter 4, we will examine how to define spaces that allow to solve high-level tasks using learning, while retaining optimisation efficiency.

In Chapter 4, we presented solutions for linking different control methods; however, these solutions relied on simplified model for the robot or a predefined division of the observation space. We extended these results by introducing a condensed FEM model of the robot, which was inspired by the one-time step control method developed by DEFROST. This model can be used for model reduction, design, and trajectory optimisation. These topics are discussed in Chapter 5.

3 RL and planning algorithms with FEM modelling

Contents

4. Trinité.

— **Nemo**

What links the different steps of life? The history of humanity and philosophy have tried to answer this question: a destiny imposed by a God, the contingency of life proposed by Sartre or the amor fati of Nietzsche, or the Ikigai in Japanese culture. It's about something that goes beyond us, or allows us to go beyond ourselves. Each individual step is a clue to a greater underlying truth. Taking a global view of different issues or problems can reveal surprising conclusions.

The first part of this work consists of directly using RL and planning algorithms on a soft-robot FEM model. This is a first step towards solving the problem of this PhD work. RL can be used to solve sequential decision problems, i.e. decision problemes over several time steps such as high level tasks. The use of RL is based on MDPs (see Section 2.2.2), which requires the definition of an observation space. In this chapter, it is based on elements of the FEM simulation (position of nodes and mechanical information).

As seen in Section 2.1.2, we worked with SOFA, which can be used to simulate a large variety of soft robots. However, it is not originally designed for sequential decision algorithms. To address this problem, we developed SofaGym (3.1.1), an API based on Gym that allows the use of any SOFA simulation for control with RL and planning algorithms. This API can be used with many robots (3.1.2) and will be used throughout this work.

The results obtained with this tool demonstrate the feasibility of using RL algorithms to solve complex tasks with soft robots, including different manipulation tasks (3.2). They also provide the opportunity to compare sequential decision and optimisation methods for solving control tasks (3.3). In addition, SofaGym allows to tackle the problem of how to define the observation space, either through the use of reduced models for the simulation and description of robots, or by combining optimisation and learning for control (3.4). These elements highlight the advantages and drawbacks of using sequential decision algorithms directly with an FEM model of soft robots. The results presented in Sections 3.3 and 3.4 are the basis of the other works in this PhD manuscript.

3.1 SofaGym, a Gym API for soft robotics

3.1.1 Introduction to SofaGym

An environment in RL includes all the elements defining a MDP. It is used to simulate interactions between an agent and an external world and to retrieve information for the learning process. The Gym framework allows to interact with an environment using a small set of simple functions, such as $step(a)$ to simulate the agent performing action a and to recover information from the environment (state, reward, flag to indicate the end of the episode), $reset()$ to restart the environment, and $render()$ to provide a visual representation of the current state. This interface provides a single generic method (API) to interact with many environments. There is no easy, complete, and open-source method to couple soft robot FEM simulations and

Gym environments, that is, to control a soft body deformed with specific actuations in a Gym environment. This is what SofaGym does.

This framework allows to use both SOFA and Gym advantages. SOFA enables to use of faithful mechanical modelling based on FEM calculations, including many soft actuators. Gym allows to manipulate an environment using a reference framework and baseline algorithms. An illustration of the API is presented in Figure 3.1.

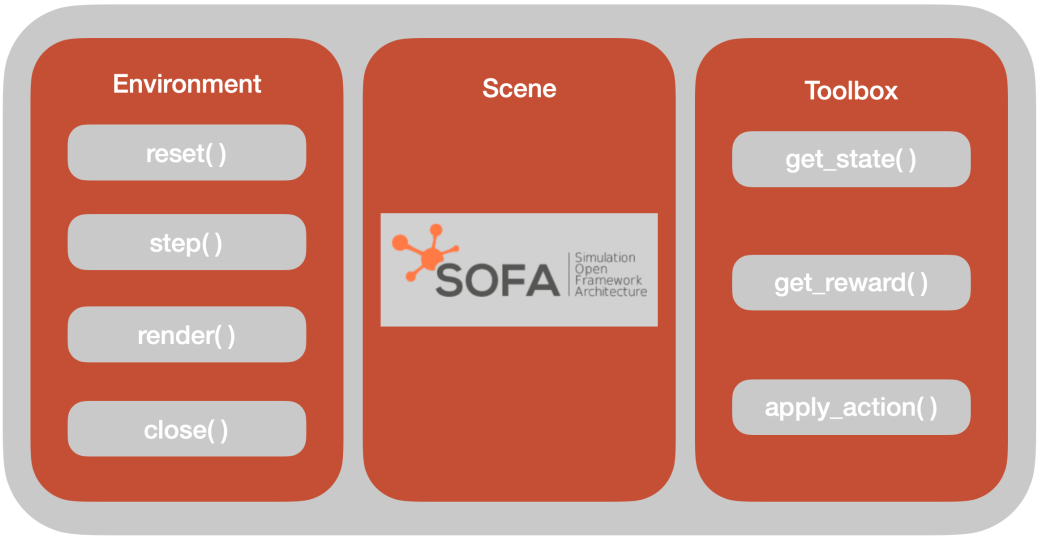


Fig. 3.1: Illustration of SofaGym components. SofaGym links SOFA (soft robot simulation) and Gym (RL environment) via a toolbox. This toolbox allows to apply in the SOFA simulation the elements given by Gym like the action and to recover the elements of interest like the observation of the robot's state or the immediate reward.

Two tools are implemented in the current version of SofaGym. To use the planning algorithms, all intermediate configurations of the environment must be saved so that they can be visited if they look promising (see 2.2.2 for more information). However, SOFA is not design to perform this step easily. Therefore, a server/worker system has been implemented. Each configuration is stored in a worker, which can be accessed by the server if necessary. To minimize the size of the manipulated SOFA scenes, we separate the mechanical model of the scene elements from the visual model. The workers contain only the mechanical models, and a viewer is used to retrieve information from these workers and display them in a visual model of the scene.

As mentioned in the previous chapter, one of the main limitations of using FEM simulations is the computational cost. SofaGym is not an exception to this rule. However, SofaGym has highlighted new research challenges. On the learning side, it is crucial to design algorithms that are as data-efficient as possible and that can learn from off-policy data. Another interesting challenge is to determine the parts of the task to be learned and how to integrate the learned elements with traditional control schemes. Many improvements can be done on the simulation side, and any increase in the simulation speed will have an immediate impact on the learning time. One approach to accelerate the simulation is to use model order reduction [GD18]. However, the results are more challenging to transfer from simulation to reality as the reduced model is more sensitive to changes in boundary conditions (particularly contact conditions).

3.1.2 Soft robots in SofaGym

It is possible to simulate different robots in SofaGym. This subsection presents the robots used in this work. We chose to separate them according to their tasks: illustrative purpose, positioning, manipulation, locomotion, and planning. The idea is to present the environments and the corresponding challenges. These environments can be used in various other settings. We will mention them when it is relevant.

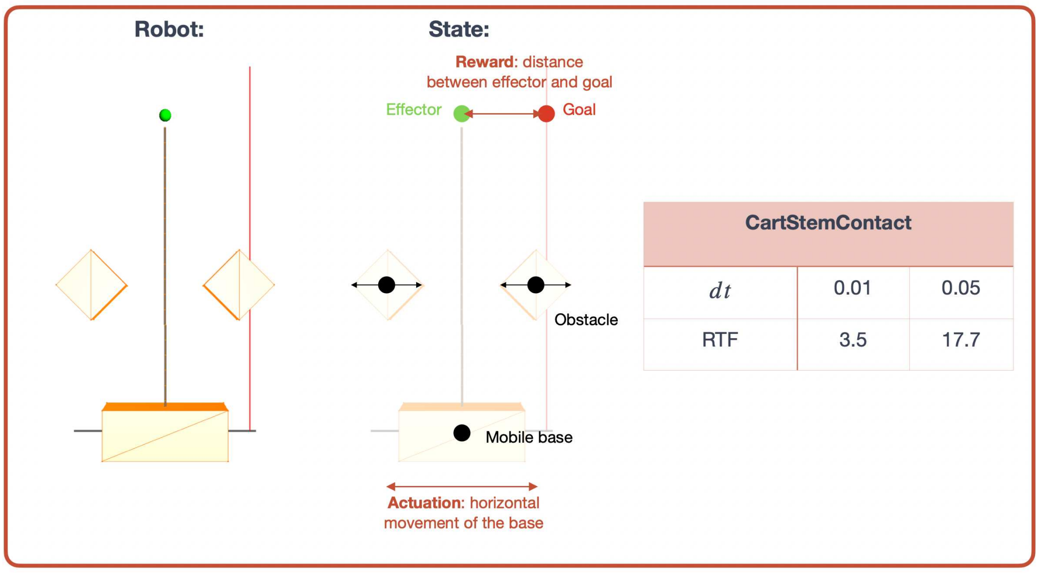
The most important technical aspect of these environments is the actual time required to perform a simulation step, which we measured using the Real-Time Factor (RTF). The RTF is calculated using the following formula:

$$\text{RTF} = \frac{\Delta t_{simu}}{\Delta t_{real}} \tag{3.1}$$

where Δt_{real} (s) is the real time required to simulate Δt_{simu} (s) of the robot behavior. If we have RTF ≥ 1, we have $\Delta t_{simu} \geq \Delta t_{real}$ and we simulate more time than the computation time. The simulation can be qualified as "real-time" if it has an RTF $= 1$. For all environments, we compute the RTF with $dt = 0.01$s. When the simulation is stable with a larger dt we also compute the RTF for this value. In fact, it is possible to improve the RTF by increasing dt, often at the cost of excessive damping (as we are using an implicit low order scheme). The mean value is computed for one episode. The RTF gives an order of magnitude of the temporal performance of the simulations. It allows for a comparison of the simulation time between different environments and with real time.

Baseline: CartStemContact

The CartStemContact environment is designed to show the importance of considering the contacts between the soft robot and objects for the resolution of a placement task. It is composed of a deformable beam attached to a mobile base. The base can translate along a defined horizontal axis called x. Two obstacles are used in the environment to block the movement of the beam. The environment and its temporal performance are illustrated in Figure 3.2. The objective is to bring the tip of the beam to an horizontal goal defined by an x_{goal} position located behind one of the obstacles. The opposite obstacle is used to bend the beam in order to reach the goal.


Robot:
State:
Reward: distance between effector and goal
Effector
Goal
Obstacle
Mobile base
Actuation: horizontal movement of the base

CartStemContact		
dt	0.01	0.05
RTF	3.5	17.7

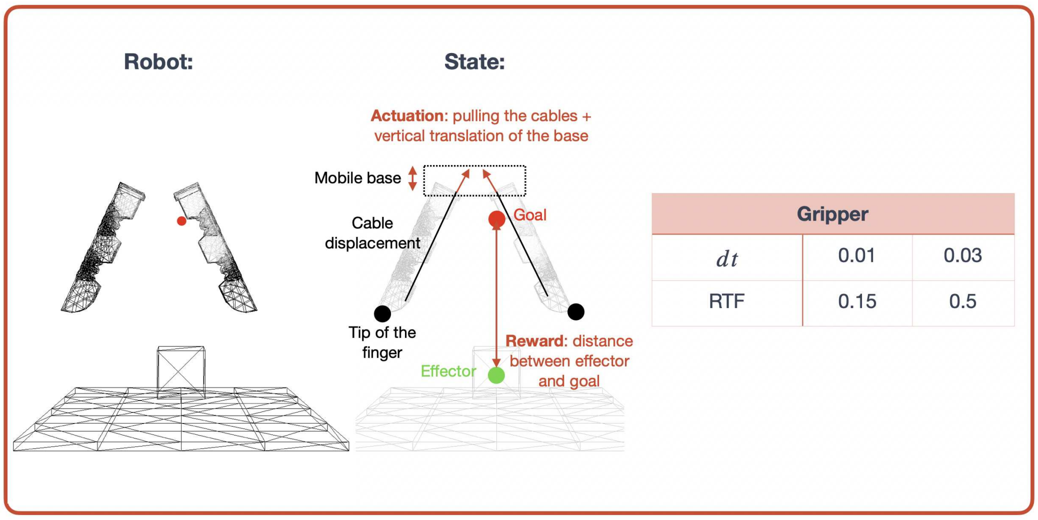
Fig. 3.2: Representation of the CartStemContact robot. Left: CartStemContact in SofaGym. A vertical deformable beam is attached to a mobile base. The base can translate along a defined horizontal axis. Two obstacles obstruct the movement of the beam. The objective is to position the tip of the beam (green point in the image) at a given abscissa (red vertical bar in the image), which may be located behind one of the obstacles. Middle: Elements used to describe the robot's state, actuation and reward. Right: RTF obtained for different values of dt (s).

The state observation is defined by the position and size of the obstacles and the position of the mobile base, the beam tip and the goal. The actuation is given by the linear movement along the x-axis. The immediate reward is the horizontal distance between the tip and the goal, expressed as $r_t = -||x_{tips,t} - x_{goal}||_2$ where $||.||_2$ denotes Euclidean norm.

Relevance of the proposed environment: The CartStemContact environment presents a problem that cannot be solved by convex optimisation, as discussed in Section 3.3.2. From a practical point of view, it provides an easy understanding of how to use SofaGym and highlights challenges that are specific to soft robotics. It serves as a baseline environment for quickly testing RL or planning algorithms in the field of soft robotics.

Manipulation: Gripper

The Gripper is a robot designed for manipulation tasks. It consists of two soft, cable-actuated fingers that can be translated in a simulated environment. These fingers have three phalanxes and are made of silicone [Nav+20]. Each finger has two pneumatic cavities, which internal pressure is measured to infer its deformations [Nav+21]. The two fingers are rigidly attached at their base, and a cable is passed through each of them. They are actuated by pulling the cable and translating the base. A manipulable cube is present in the environment, and forces with random magnitude and direction are applied to test the grip robustness. The environment and its temporal performance are illustrated in Figure 3.3. The objective is to grasp a deformable cube and bring it to a certain height, with a greater reward the closer the cube is to the target.


Robot:
State:
Actuation: pulling the cables + vertical translation of the base
Mobile base
Cable displacement
Goal
Tip of the finger
Reward: distance between effector and goal
Effector

Gripper		
dt	0.01	0.03
RTF	0.15	0.5

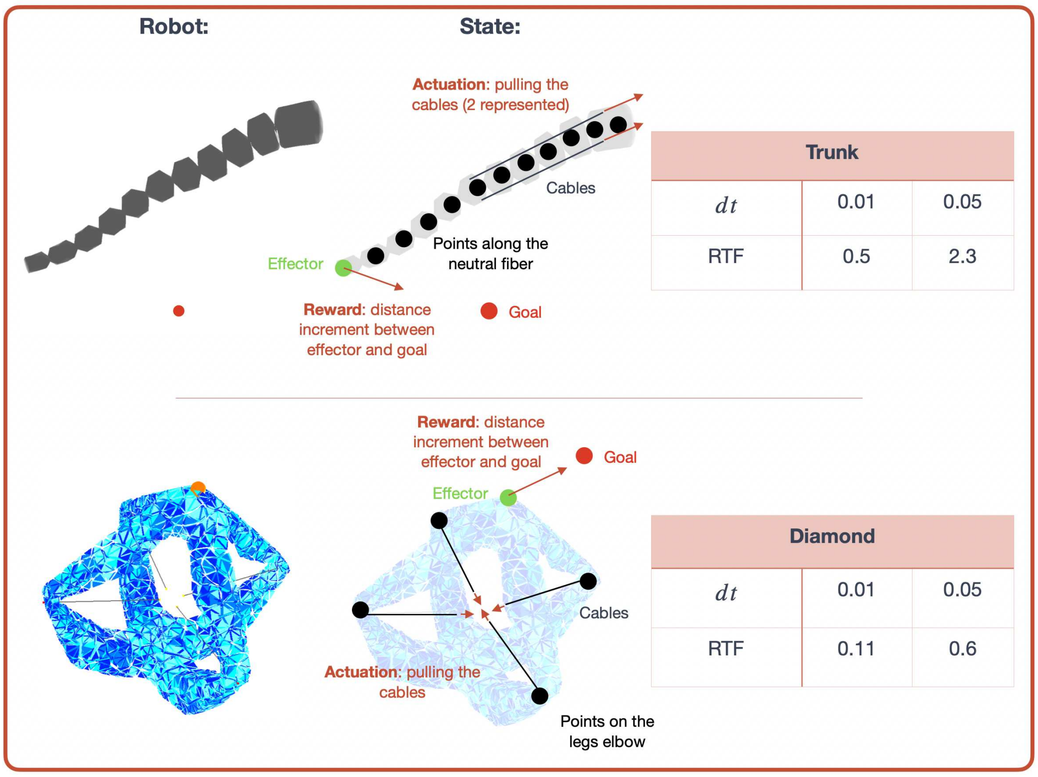
Fig. 3.3: Representation of the Gripper robot. Left: Gripper in SofaGym. A cable-actuated gripper is used to bring a deformable cube to a certain height, as indicated by the red point in the figure. Middle: Elements used to describe the robot's state, actuation and reward. Right: RTF obtained for different values of dt (s).

The state observation is defined by the object and goal positions, the displacement of the cable, the tips of the fingers. The observation is normalized by the extreme values that could be obtained. The actuation can be either discrete or continuous and is given by the translation of the base along any axis and the displacement of the cables. Let z_{cube} be the z-coordinate of the position of the center of mass of the cube. The immediate reward is calculated as the vertical distance between the cube and the goal, expressed as $r_t = ||z_{cube} - z_{goal}||_2$.

Relevance of the proposed environment: Soft grippers are an important part of the soft robotics field, and have already been transferred to the industrial world. They remain challenging because of the contact between the gripper and the objects to be grasped. Soft grippers are excellent examples of the adaptability of soft robots. The fingers of the gripper adapt to the shape of the object being grasped when sufficient force is applied. The ideal reward considers this adaptability.

Positioning and handling: "Trunk" and "Diamond"

The objective of a manipulation or positioning task is to place the effector of the robot or the manipulated object at a specific position and orientation, which requires learning the inverse model of the robot. The so-called « Diamond robot » [Dur13] is a soft parallel robot built with silicone and operated using four cables anchored in the middle of each leg and pulling towards the robot's center. The so-called "Trunk robot" [CED17] is inspired by an elephant trunk, and composed of a slender flexible silicone body along which eight actuating cables are guided. The robot consists of two segments that can be bent using two pairs of antagonistic cables per segment. These environments and their temporal performance are illustrated in Figure 3.3.


Robot:
State:
Actuation: pulling the cables (2 represented)
Cables
Points along the neutral fiber
Effector
Reward: distance increment between effector and goal
Goal
Reward: distance increment between effector and goal
Goal
Effector
Cables
Actuation: pulling the cables
Points on the legs elbow

Trunk		
dt	0.01	0.05
RTF	0.5	2.3

Diamond		
dt	0.01	0.05
RTF	0.11	0.6

Fig. 3.4: Representation of the Trunk robot and Diamond robot. Left: Trunk (up) and Diamond (down) in SofaGym. The Trunck is actuated such that its tip reaches a goal, represented in red in the figure. The tip of the Diamond robot, in orange in the figure, must be brought to a certain position. Middle: Elements used to describe the robot's state, actuation and reward. Right: RTF obtained for different values of dt (s).

For the Trunk robot, the observation is defined by a set of points uniformly distributed along its neutral fiber and the position of the goal. Actuation can be either discrete or continuous and is achieved by changing the lengths of the eight cables. The immediate reward is calculated as the normalized positive decrease in the distance between the tip and the goal. Let p_t be the position of the robot's tips at time t and p_{goal} be the position of the goal. The immediate reward is given by the following equation:

$$r_t = max(0, \frac{||p_{t-1} - p_{goal}||_2 - ||p_t - p_{goal}||_2}{||p_{t-1} - p_{goal}||_2 + \epsilon}) \tag{3.2}$$

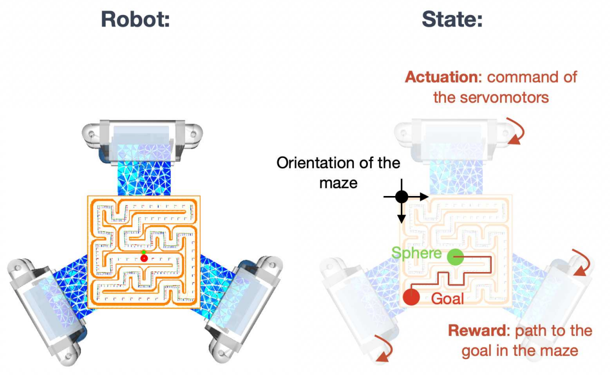
where ϵ is a small quantity used to prevent division by zero. By dividing by the previous distance, we obtain a normalised value for the variation in distance from one time step to the next.

The Diamond robot's observation is defined by the positions of its top point and leg elbows. Its actuation can be either discrete or continuous and is provided by the length of its four cables. The immediate reward is the same as that specified in Equation 3.2 for the Trunk.

Relevance of the proposed environment: These two environments show that the inverse model can be learned through training or planning methods; however, this requires considerable training time and samples. These environments are intended to showcase the value of inverse model control and to encourage research on how to combine learning methods with existing control schemes rather than replacing them entirely.

High level control example: Maze and SimpleMaze

The Maze environment includes a small rigid sphere, a rigid maze, and a Tripod soft robot [Van+20]. The Tripod robot consists of an elastic body with three legs, each actuated in rotation and translation by a servo motor attached at its base. The maze is fixed to the Tripod, and deforming the silicone structure changes its orientation. The sphere moves when the orientation of the maze is changed, the goal being to reach one of the four corners. Two versions of this environment are available: one composed of the maze only (SimpleMaze), allowing for direct orientation to move the sphere, and another that includes the maze and Tripod (Maze), with the orientation of the maze determined by the movements of the Tripod. The environments and their temporal performances are shown in Figure 3.5.


Robot:
State:
Actuation: command of the servomotors
Orientation of the maze
Sphere
Goal
Reward: path to the goal in the maze

Maze		
dt	0.01	0.05
RTF	0.02	0.06

SimpleMaze		
dt	0.01	0.05
RTF	1.26	9.0

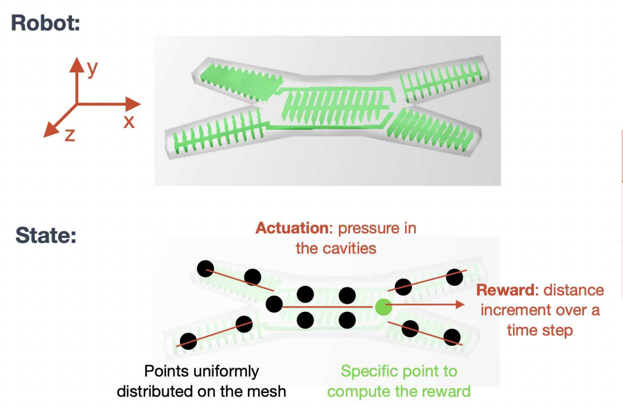
Fig. 3.5: Representation of the Maze environment. Left: Maze in SofaGym. A Tripod soft robot is employed to orient a maze to position a sphere to a specific location. The SimpleMaze environment is simply the maze that can be oriented. Middle: Elements used to describe the robot's state, actuation and reward. Right: RTF obtained for different values of dt (s).

For both versions of the environment, the observation is defined by the orientation of the maze, the position of the sphere and the position of the goal. In the complete Maze environment, the actions are the command of the servomotors. In the simplified version, the actions orient the maze directly. The reward is determined by the shortest path to the goal, calculated using a Dijkstra algorithm [Dij59].

Relevance of the proposed environment: These environments are designed to investigate the interaction between RL and planning and traditional control schemes. The inverse model of the Tripod robot can be solved using optimisation methods. Therefore, the maze can be solved using planning or learning methods, and the inverse model can then be computed to determine the servomotor positions associated with these maze orientations. This separation is also justified by the computation times of the two environments.

Locomotion example: Multigait

The Multigait robot [She+11] is a crawling soft robot. The goal is to move forward in a given direction x as fast as possible. The robot is actuated by inflating or deflating five cavities, one in the center and one in each leg. The environment and its temporal performance are illustrated in Figure 3.6. The robot is build according to geometric symmetry planes, and control strategies can exploit these symmetries to achieve specific movements. For example, symmetry along the plane x-y defined by the main axis and vertical axis can be used to move forward, which results in the same actuation of the front legs (resp. the back legs).


Robot:
y
x
z
State:
Actuation: pressure in the cavities
Reward: distance increment over a time step
Points uniformly distributed on the mesh
Specific point to compute the reward

Multigait		
Modelling	Complete	MOR
RTF	0.007	0.15

Fig. 3.6: Representation of the Multigait robot. Top left: Multigait in SofaGym. The robot moves along the horizontal axis. MOR can be used to simulate the robot. Bottom left: Elements used to describe the robot's state, actuation and reward. Right: RTF obtained for the complete and the reduced model of the robot.

To improve the performance of the MultiGait simulation, Model Order Reduction (MOR) is used [GD18]. This method is based on the projection of a simulated model into lower-dimensional space. Proper Orthogonal Decomposition (POD) is used to find the translation and deformation modes that describe at best all possible configurations of the robot. These deformation modes are obtained from a database of possible robot configurations, generated in simulation by discretising the possible actuation values and solving the direct model. In the case of the Multigait, 63 modes are used (3 rigid translational modes and 60 modes of deformation sorted from the most to the least important). Using this reduced model, we can increase the RTF as shown in Figure 3.6. The reduced model in the corresponding environment is called ReducedMultigait.

For the complete model, the observation is defined by a finite set of points uniformly distributed on its mesh. For the reduced model, the observation is defined by the intensity of the deformation modes. In both cases, the actuation is pneumatic and can be loaded continously or with discrete values. The robot has 5 pressurised cavities. Let x_t be the position along the privileged axis of one of the points of the robot (e.g., the central point). As the episode has a fixed time length, the immediate reward is defined by the position increment made in one time step, which is written $r_t = x_t - x_{t-1}$.

Relevance of the proposed environment: MOR provides an explicit mechanical representation of the robot through deformation modes. In addition to improve the simulation time of the robot, this model allows to describe the robot's state using a low-dimensional variable (e.g. 32 deformation modes) while retaining mechanical informations. The objective is to determine how to leverage this mechanical model of the robot to support learning. As the Multigait robot is a well-known example in the soft robotics community, it is also possible to compare the results obtained in the simulation with a hand-crafted baseline. Finally, it is possible to evaluate the learning transfer between the two environments.

3.2 Solving manipulation tasks with RL

The presented SofaGym and robots are now exploited. First, we concentrate on two examples of manipulation, as this is a significant challenge in soft robotics. Gripper and Trunk environments can be employed to perform complex manipulation tasks. The first example involves manipulating a deformable cube using the Gripper robot, showcasing the feasibility of using learning methods such as Behavior Cloning (BC) to solve a task with a FEM simulation. The second example is the handling of space debris using the Trunk robot, demonstrating the possibility of defining task-specific rewards using biomimicry. These tasks cannot be solved directly with one-timestep control tools, as it is necessary to define a trajectory for approaching the object before it can be manipulated.

3.2.1 Manipulation of a deformable cube with the Gripper robot

The problem is reformulated as a control policy [Ope+19b] to pick up the cube and lift it to a specific goal height. To solve this task, we use the PPO algorithm, with a strategy represented by a recursive network (LSTM) with 128 neurons. This algorithm successfully converges after $1.4e6$ iterations. However, the grip is weak as the gripper only moves one finger and uses the second finger as a support. To encourage firmer grips, a random force that pulls the cube down is added at each time step. This makes the gripper firmly hold the cube, as it can maintain the cube and continue moving without any issues. Both cases result in the discounted sum of rewards converging to the same value, which corresponds to the task resolution. The results are presented in Figure 3.7.

Behavior Cloning (BC) is used to speed up the training process. BC treats the problem of Imitation Learning as a supervised learning problem by utilizing expert trajectories (observation-action pairs) to train the policy network to replicate expert behavior. It is required as a manual input from the user. In the Gripper case,

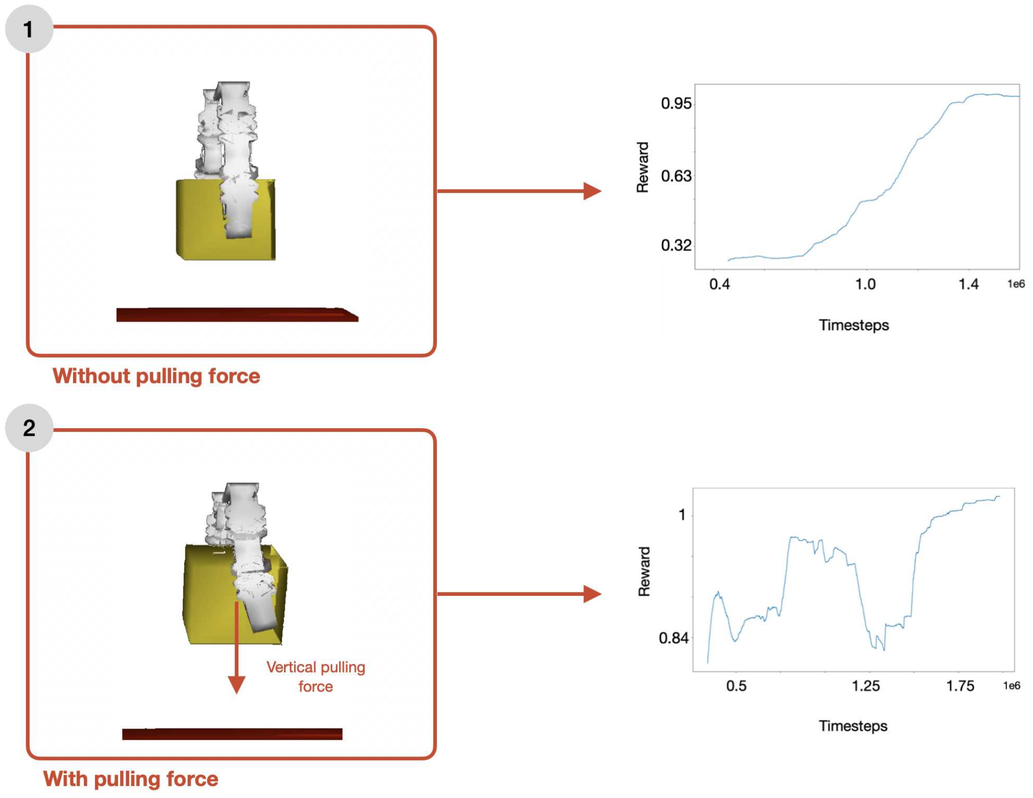


Fig. 3.7: Grasp learned without (1) and with (2) vertical pulling force. The addition of force shows that the gripper is able to adapt and use both fingers to effectively catch the cube. This is confirmed by the curve rewards against the time step. Rewards are normalized to the largest value of the plot

the trajectory represents a firm grasp that moves the cube to the desired height. The policy network is pretrained on these data for 3000 epochs before running the standard PPO algorithm. Pre-training both accelerates the learning process and guides the agent to successfully learn the expert behavior, that is to center the gripper before grasping and lifting the cube. These elements are shown in Figure 3.8. An extension of this work could be to learn how to manipulate any object from its 3D mesh.

3.2.2 Handling space debris with a bio-inspired reward

Both RL and the use of soft materials allow the soft robot to adapt to its environment. The mechanical adaptations experienced by the body are used to facilitate the learning. Therefore, the reward function has to be designed to consider this body's adaptation. We illustrate this by using a soft trunk to capture space debris (see Section 3.1.2 for a description of the Trunk robot). This choice is based on the increasing number of man-made objects in space and the need for space debris removal missions. Because space debris are not designed to be caught, soft robotics offers the advantage of simplifying the grasping of irregular objects. We consider

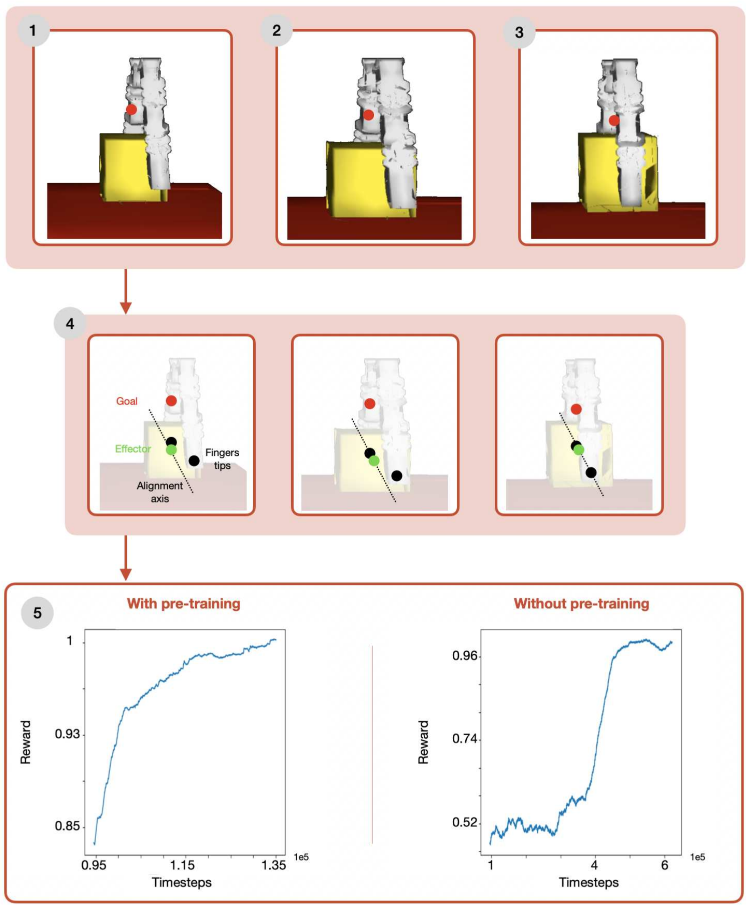


Fig. 3.8: Learned Policy with BC. A hand-designed strategy is implemented to center the gripper on the cube before catching it. The learning process begins with this strategy and generalizes it to different positions of the cube. The gripper follows a sequence of four steps: starting from an initial configuration (1), centering the first axis (2), centering the second axis (3), and moving the caught cube. These different steps are based on the definition of key points on the robot (tips of the fingers) that have to be aligned to hold the cube efficiently (4). These steps allow to easily define a sequence to position the fingers around the cube without touching it, and then to catch it. Training is accelerated by using BC (5). Rewards are normalized to the largest value of the plot.

spatial conditions (ignoring gravity) and a cubic object to be manipulated, as shown in Figure 3.10.

The reward function is obtained based on the robot' interactions with its environment by defining the contact points that force it to bend. Our inspiration comes from the ability of an elephant to perform dexterous grasping tasks with its trunk. The proximal section of the trunk (from the elephant's head to the middle section) is used for positioning, whereas the distal section (from the middle section to the tip) is mainly used for dexterous tasks. We formulated the whole-arm grasping according to this behavior. Let T be a point on the proximal section, P be the middle point of the trunk, D be the point at the tip, M and N be two points on opposite faces of the cube, as shown in Figure 3.9. The learned grasping sequence that allows the bending of the trunk and the grasping of the cube consists of a proximal approach phase, where the distance $||\vec{PM}||$ is minimized, a primal deformation, where the distance $||\vec{DN}||$ is minimized, and a wrapping phase, where the center of the cube gets closer to point T.

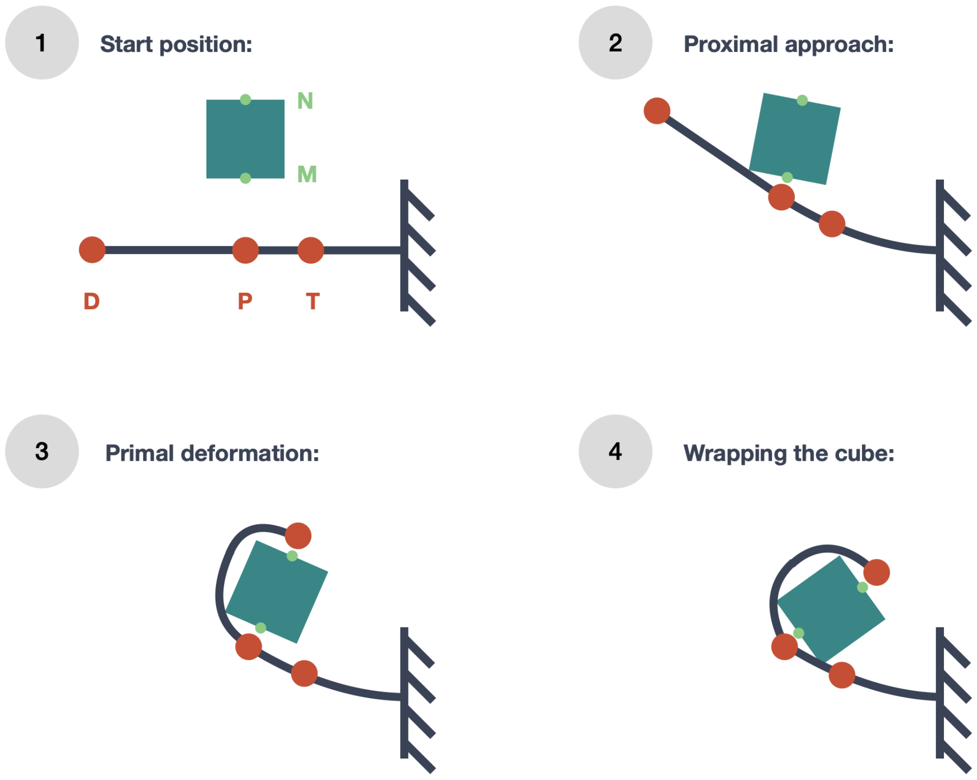


Fig. 3.9: 2D schematic of the whole-arm strategy. 1) Start position: contact points on the trunk robot (D, P, T) and object (M, N). 2) Approach: approaching the object by minimizing the distance $||\vec{PM}||$ between the proximal part of the trunk and the object. 3) Deformation: minimizing the distance $||\vec{DN}||$ between the distal part of the trunk and the object by bending the distal part. 4) Wrapping: wrapping the trunk robot around the object by pushing it towards point T.

The immediate reward r_t incorporates the whole-arm grasping strategy according to:

$$r_t = -\frac{||\vec{PM}_t||}{||\vec{PM}_0||} - \frac{||\vec{DN}_t||}{||\vec{DN}_0||} - \frac{||\vec{CT}_t||}{||\vec{CT}_0||} \tag{3.3}$$

The distance is calculated using the Euclidean norm and normalized by the initial distances between the points to eliminate the dependence on the initial relative position of the trunk and the cube. A negative sign is used to find the fastest strategy. The cube's dimensions and position are added to the observation, and the actuation is continuous. We suppose that the grasping is successful when the trunk achieves distributed contacts, and the cube is kept close to it. At the beginning of each new learning episode, the size of the cube and its position are randomized, always within the grasping capabilities of the Trunk robot.

To solve this task, we use the PPO algorithm, with a strategy represented by an MLP network of three layers of 512 neurons each. This lead to successfully learn the whole-arm manipulation strategy after $1e5$ iterations, as illustrated in Figure 3.10. Over 400 tests with different combinations of cube position and size, the success rate is 80%, with failure due to dropping the cube after catching it in 70% of the failure cases (small-sized cube, ill-defined end of the episode). The strategy is validated using cube positions and sizes that were not included in the training set. The limitations of this method are two-fold. First, the opposite face defined during training may not be the opposite face when the trunk catches the cube, especially when the object starts to rotate after being hit by the Trunk. Second, the reward is defined as the distance between points, which causes the Trunk to place the tip on one side (one point) instead of using a wider area to ensure firmer grip. Increasing the number of contact points, changing their location, or allowing different contact positions on the cube would address these issues.

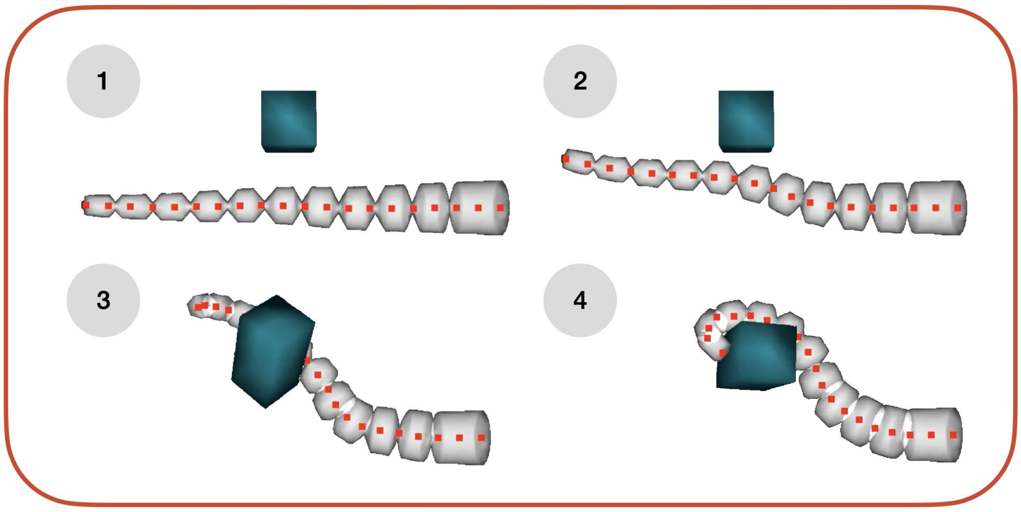


Fig. 3.10: Whole-arm strategy implemented by the Trunk robot to grasp a cube.

3.3 Comparison of learning and optimisation control methods

In Chapter 2, we discussed the distinction between control methods that involve learning and those that involve optimisation. Soft robots can be controlled using traditional methods such as open-loop inverse control based on mechanical modelling. These methods do not require a learning time but are limited by strict mathematical assumptions about the optimisation domain. On the other hand, RL and planning are capable of handling problems without these strong assumptions about the spaces, making them more suitable for problems where contacts lead to changes in the optimisation domain. Each of these methods is therefore based on a different way of describing the state of the soft robot (condensed model, state observation), which leads to different limitations. The objective of this section is to illustrate these differences using different types of soft robots and tasks. The idea is to understand the limitations and strengths of both approaches.

3.3.1 Performing a positioning task

Some of the tasks presented in SofaGym have already been solved using optimisation. This is the case of the positioning task with the Trunk and the Diamond robots presented in Section 3.1.2. To control these robots, we use both planning algorithms (see Section 2.2.2) and a one-step optimisation scheme (see Section 2.2.3). The idea is to compare the performances of these two methods in a task in which the goal is to move the robot's effector to a predefined target position. This task can be classified as a low-level control task. We used QP to refer to the specific algorithm for the optimization based inverse model, and the OPD algorithm for planning. Both QP and OPD have the same positioning performances. Due to the need to construct the planning tree at each planning step, the OPD takes approximately 10 times longer to solve the same trajectory. The results for the Diamond are presented in Table 3.1.

Algorithm	**Mean Computation Time** (s)
QP	3.93
OPD	38.24

Tab. 3.1: Comparison of the computation time using QP and OPD to solve a positioning task with the Diamond robot.

In the case of the Diamond robot, we observe that there is only one solution for reaching a given position. If there are more actuators than end-effectors (taking into account their dimension), the optimisation can lead to a local minimum. In the presence of contact, the problem becomes non-convex. In addition, one time step methods are based on the linearisation of the mechanical model. Learned approaches require a long period of training, which is not the case for optimisation approaches such as QP. In general, optimisation approaches are associated with closed loops to correct model errors. It is possible to prove that optimisation-based control and closed loop converge, which is not as easy with learned models. Learned methods can theoretically lead to similar or even better performance than optimisation-based approaches, particularly in terms of accuracy when the problems are no longer convex. These examples demonstrate that not every problem should be solved using learning or planning methods. One of the main challenges in applying RL or planning in soft robotics is to use joint traditional control schemes and learning tools, combining the advantages of both approaches.

3.3.2 Contacts and optimisation space

In soft robotics, when the contact configuration changes, the robot moves from one optimisation space to another and the optimisation problem changes. This concept is easily illustrated using the CartStemContact example presented in Section 3.1.2. The resolution of this problem with QPCC (QP with complementarity conditions, to deal with contacts) algorithm is depicted in Figure 3.11.

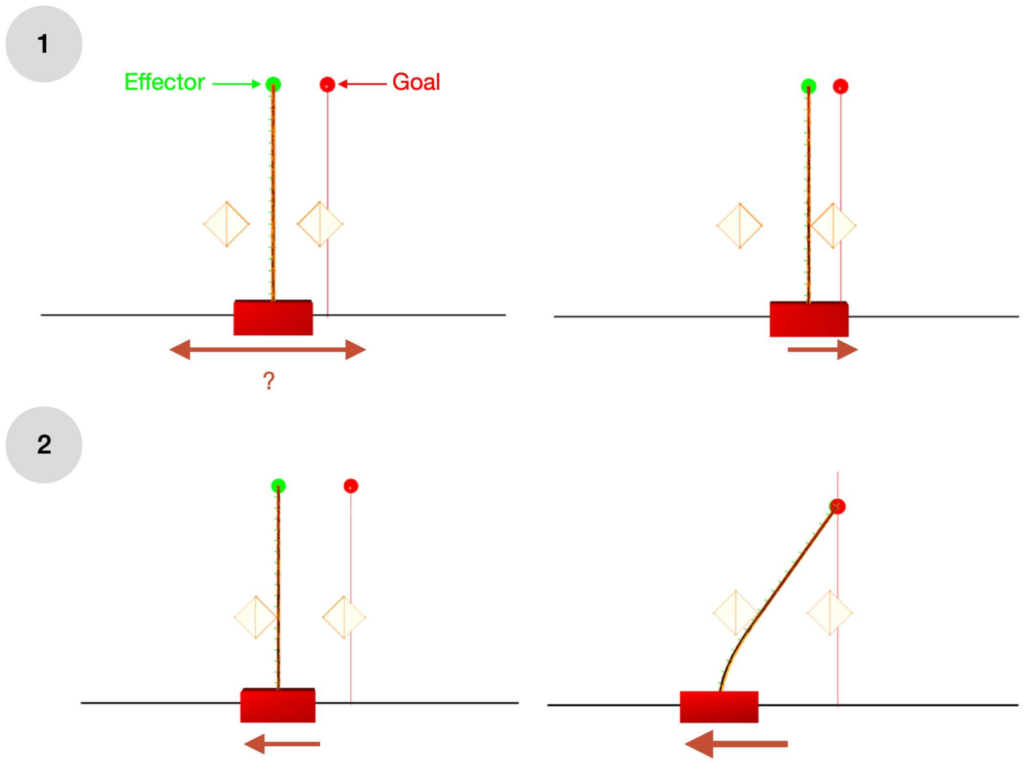


Fig. 3.11: Example of solving a positioning task with CartStemContact and QPCC algorithm. 1) When the robot starts without contact, the QP algorithm falls into a local minimum. If the cart moves to the right, it creates contact and moves away from the goal; if it moves to the left, it also moves away from the goal (top right). 2) To solve this task, the beam must first be in contact with the opposite obstacle.

This environment exhibits three possible contact configurations: contact on the left of the beam, no contact, and contact on the right of the beam. Let the goal be located behind one of the obstacles and assume that the beam is in the no-contact configuration. From this configuration, the optimisation problem minimizes the distance between the goal and the tip of the beam by moving the cart toward the obstacle, resulting in a local minimum. In fact, the QPCC solve the problem in the "no contact" optimisation space, and get stuck when the optimisation space changes. To reach the goal using the QPCC, the beam has to be in contact with the opposite obstacle. In contrast, RL algorithms do not have the same limitations. The task can be solved after $2e5$ iterations using SAC algorithm with a strategy represented by an MLP network of three layers of 512 neurons each. For this example, we consider an episode with 30 time steps of 0.3 seconds each. The episode stops when the timer is reached or when the distance falls below a specific threshold, set at 0.15 cm (less than 1% of beam length). The initial positions of the goal and cart are random. The cumulative reward and the strategy found by the algorithm are shown in Figure 3.12. SAC can overcome the convex space breaks imposed by contacts, which is not possible with convex optimisation methods. The advantages of RL algorithms are twofold. When the environment is stochastic, it allows for generalization and obtaining a robust controller. When the environment is deterministic, the intrinsic exploratory aspect of these algorithms enables them to find trajectories to solve a task when optimisation techniques, such as QP, do not work.

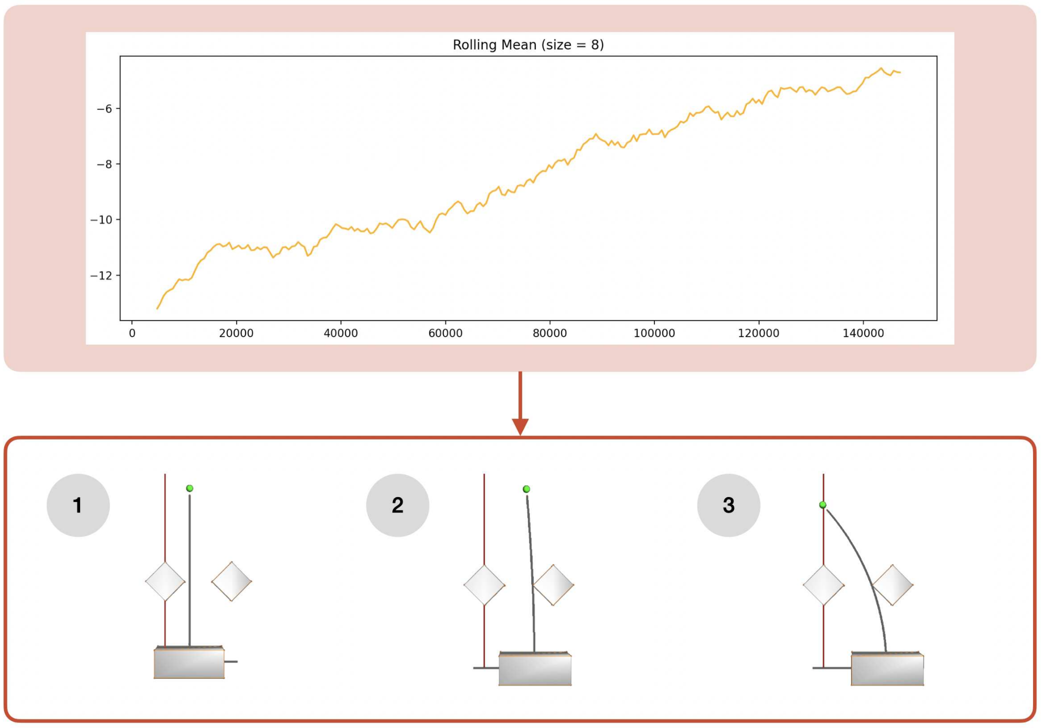


Fig. 3.12: Solving the CartStemContact problem using the SAC algorithm. Top: cumulative reward. Bottom: solving steps. 1) Random goal and base positions. 2) Using the opposite contact to deform the beam. 3) Solving the positioning task.

This topic is related to the issue of contact reconfiguration. Specifically, it can be illustrated using the Trunk robot that handles objects, such as a cup, as shown in Figure 3.13. In this scenario, QPCC allows for a real-time solution of positioning and orientation problems. However, finding a way to wrap the trunk around the cup in a manner that creates a strong grasp, good controllability, and allows for a large range of motion for the cup is not a straightforward task. This is because the position in which the Trunk catches the cup influences its movements. If the goal is not within reachable positions with this trunk configuration, the QPCC will remain stuck at a local minimum. Manipulating the cup with the QPCC and changing the grasping, if necessary, appears to be a promising research opportunity.

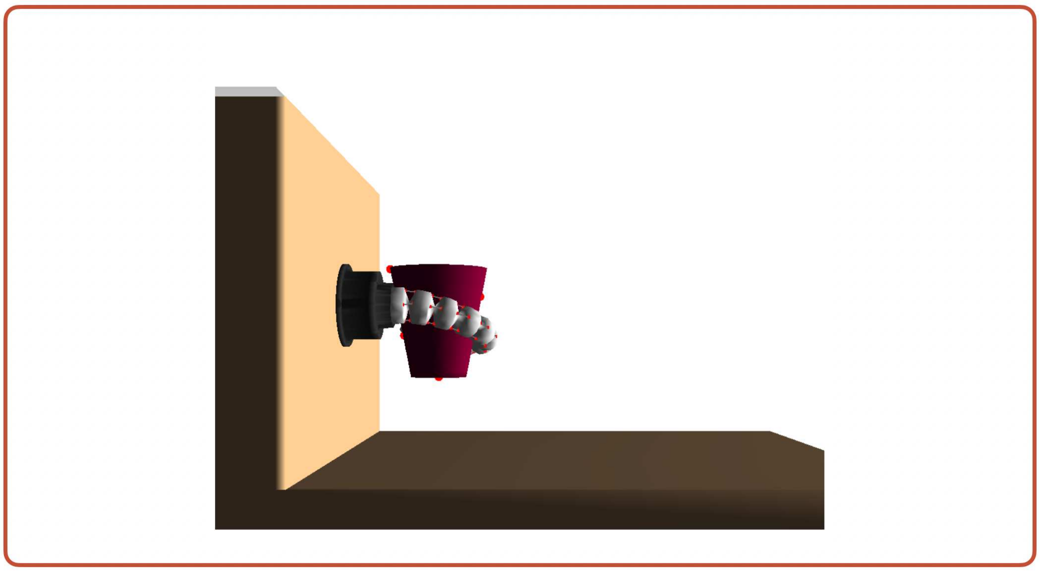


Fig. 3.13: Illustration of the Trunk robot handling a cup.

3.4 Highlighting the importance of the soft robot description using SofaGym

SofaGym provides also a platform for exploring the concept of soft robot state description. The Multigait environment (see section 3.1.2) and the Maze environment (see section 3.1.2) are available in two versions. These variations are based on different simulation models. The goal of this section is to determine how these robot descriptions can be used to solve the underlying task.

3.4.1 Using a reduced model with the Multigait

MOR allows to simulate the Multigait robot faster than FEM model (see Section 3.1.2) while retaining the most essential deformation modes to describe its behavior. This reduced mechanical FEM model can be used as observation in a RL problem. Our objective is to learn a strategy that moves along the x-axis as efficiently as a hand-defined strategy based on this observation. This hand-defined strategy involves inflating the back legs, then the central cavity, then the front legs, and finally deflating in the same order, as shown in Figure 3.14.

In the ReducedMultigait environment, the weight of each translation and deformation mode in the current configuration of the robot is a compact description

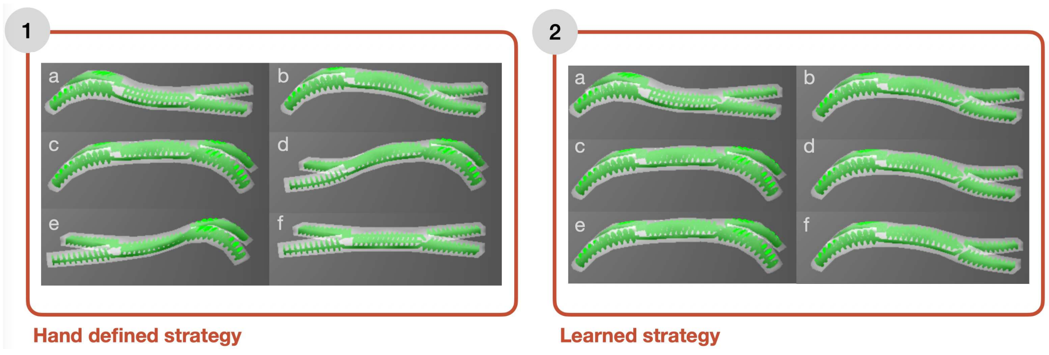


Fig. 3.14: Hand-defined and learned strategy steps for solving the locomotion task with the Multigait robot.

of its state. In our experiment, the 32 most significant deformation modes are used for observation. However, the accuracy of the simulation is limited to the scope of the model. For example, there are no "rotation modes" in the ReducedMultigait robot model, so these modes cannot be used during simulation or training. Asymmetric actuation leads to inaccurate results because it involves rotations. The action space is limited to symmetrical actuation, such as the joint actions of the two front and joint actions of the two back legs. An episode is 6 time steps long like the duration of the hand-defined strategy. To solve this task, we use the PPO algorithm, with a strategy represented by an MLP network of three layers of [256, 128, 64] neurons respectively. This lead to successfully learn the strategy after $1e4$ iterations, as shown in Table 3.2. The learned strategy is more efficient than the hand-defined strategy, and uses the front legs of the robot to move forward, as shown in Figure 3.14.

Strategy	**Cumulative Reward** (cm)	**Final x position** (cm)	
		Reduced model	Complete model
Hand Defined	5.38	3.24	3.56
Learned	6.33	4.51	6.21

Tab. 3.2: Comparison of the cumulative reward and final x position, for complete and reduced model, obtained with hand defined and learned strategy, for an episode of 6 time steps.

These results show that MOR is a feasible approach to reduce the dimension of the observation in a convenient and automatic way while still retaining sufficient information to learn an efficient strategy. The strategy discovered with the reduced model remains valid for the complete model, and it is obtained 21 times faster than if we use the complete model for learning. However, the numerical results may differ due to differences in the boundary conditions, as the reduced model eliminates local deformations that are important for contact. The results are only valid for the considered models, and an experimental validation is required to test the real efficiency of the learned strategy compared to a hand-defined strategy.

3.4.2 Planning the resolution of the maze and then controlling the robot with an inverse model

As explained in Section 3.1.2, the Maze environment allows to separate the tasks of solving the maze and controlling the Tripod orientations to combine the advantages of both planning and inverse modeling. The OPD algorithm and SimpleMaze environment are used to plan a trajectory in the orientation space, which allows the ball to move from its initial position in the center of the maze to each of the four corners. The inverse model of the Tripod robot then follows this orientation trajectory and converts the maze orientation into servomotor positions. As the Tripod end effector and maze are rigidly linked, controlling the orientation of the effector is equivalent to controlling the orientation of the maze.

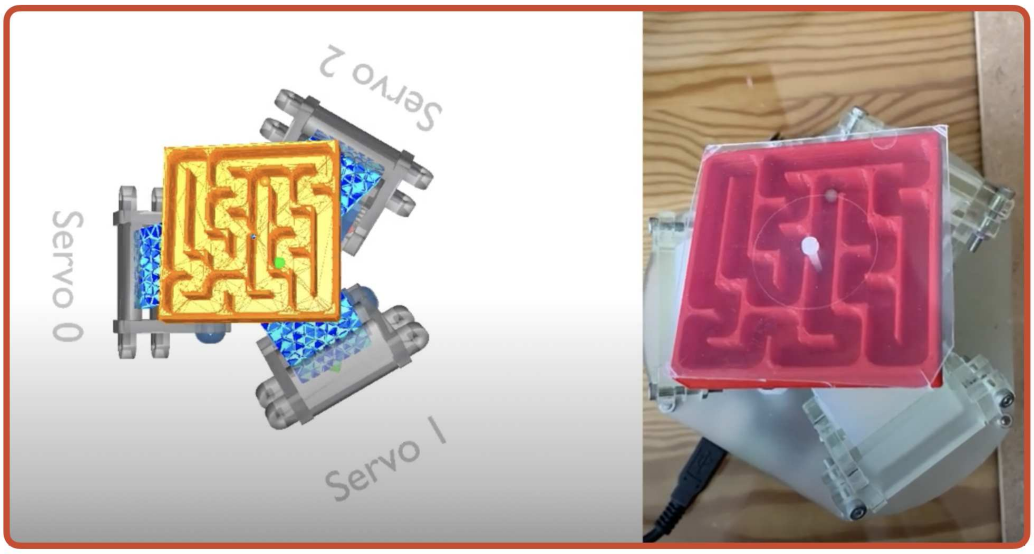


Fig. 3.15: Solving the maze in simulation with inverse model and in real world with open-loop control.

By using these servomotor positions, we can control the physical Tripod to execute the action sequence in an open-loop manner and solve the maze with the Tripod robot in the real world. An illustration of the last two steps is presented in Figure 3.15. This example is a simple demonstration of how a learned policy can be transferred from a simulated environment to reality, in the case of simple dynamics and a long time step between actions to allow the ball to settle in a position before taking the next action. Despite being a simple case, it still illustrates how a learned controller can be used in conjunction with more traditional robotics tools to learn a task in simulation and transfer it to reality.

3.5 Conclusion

In this chapter, we have shown that it is possible to use planning and RL algorithms to solve positioning, manipulation, and locomotion tasks. Planning and RL methods allow to solve tasks that cannot be solved with optimisation methods, especially when the task requires a reconfiguration of contacts as in the CartStem-Contact environment. However, RL considers robot dynamics as a black box function, and its exploratory behavior produces 0-order estimates of the gradient of the cost function leading to large variances in the gradient and a slower optimisation process. In cases where there are no stochastic effects or discontinuities in the optimisation space, optimisation methods are preferred. This is what we show with the Diamond and Trunk examples: planning and optimisation methods lead to the same results, but the optimisation methods are more efficient in the sense that they do not need a training phase or the construction of a plan to work. If there is uncertainty in the modelling task or some configurations that are not handled by optimisation methods, RL helps to increase the robustness of the controllers.

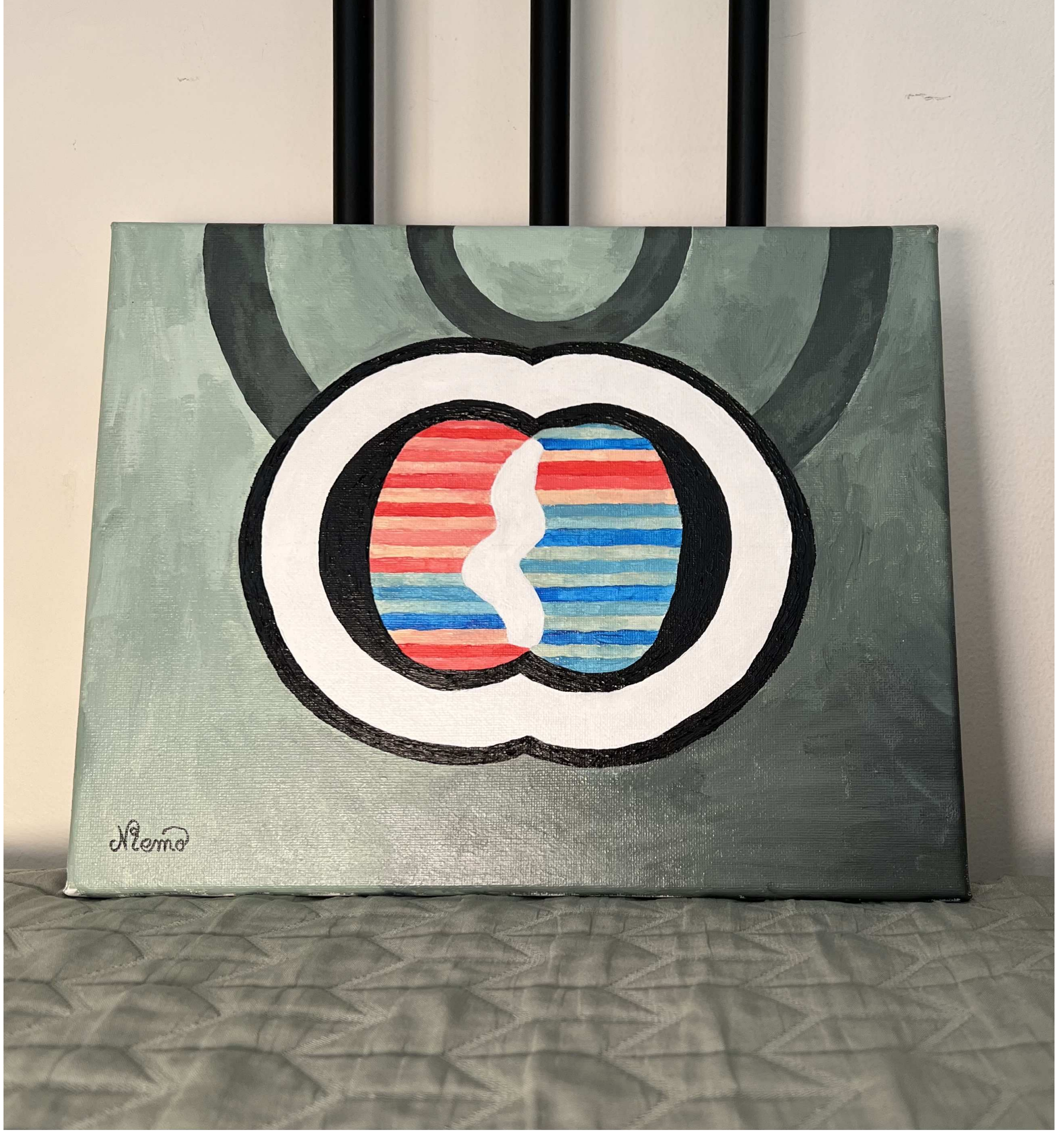
One of the main limitations of using FEM with learning approaches is finding a good trade-off between computation time and accuracy. In the previous examples, the large amounts of simulated data required to obtain an efficient and stable control policy (when it is obtained) are challenging to manage. It is beneficial to have a framework such as SofaGym, which allows adjusting this trade-off. Additionally, using this framework has enabled the development of original approaches, such as combining learning with inverse models or reducing the size of the models. These approaches differ in the description of the robot, with the ReducedMultigait environment using a reduced mechanical model to simulate the behavior of the robot and describe its state and the SimpleMaze environment focusing on high-level task resolution. The goal is to learn high-level control and perform low-level control by using known and mastered techniques.

This chapter highlighted the strengths and limitations of various modelling and control approaches in soft robotics and explored how they can be combined to achieve efficient solutions to these challenges. This serves as the basis for Chapter 4 and Chapter 5,where these approaches are combined by focusing on the description of the state of the soft robot.

4

Coupling learning and optimisation for soft robot control

Contents



5. Symbiose.

— **Nemo**

When two souls connect, they intertwine, and sometimes merge. One becomes part of the other. What emerges is something new. Sometimes, this entity lasts, and sometimes it vanishes. At the point of symbiosis, there is an unknown that we cannot fully grasp until we observe it. This also holds true for the ideas. Sometimes two ideas seem different - beautiful separately, but never explored together. However, the connection between these ideas creates new ideas, which, in turn, creates a perfect junction between the two.

Chapter 3 demonstrated the feasibility of using both RL/planning algorithms and convex optimisation methods directly on the FEM simulation of a soft robot to solve various tasks, such as positioning, manipulation, and locomotion. There are advantages and disadvantages to each method, and the literature opposes them in most cases. This chapter brings together two contributions that combine the advantages of optimisation and learning-based control approaches. Both approaches are based on a particular description of the state of a soft robot, which is why they have been grouped together in the same chapter. They build on and extend two results highlighted by SofaGym.

The Multigait example showed that it is possible to simulate the robot using a reduced model and to use information from this reduced model to represent the state of the robot. Additionally, the observation of Trunk and Diamond robots is based on the FEM mesh and uses interest points that reflect their configuration. These various descriptions have a common objective: to capture the state of the soft robots in their configuration space to use this information for learning or planning. However, not all the simulated information is used to describe the robot state, which results in unnecessary calculations. Inspired by humanoid robotics (see Section 2.4.2), Section 4.1 presents a motion generation method based on a simplified model of a soft robot, obtained by considering a reduced configuration space. Furthermore, this approach follows the statement defined in Section 3.4.2, which consists of learning a sequence of robot configurations and using inverse modelling and convex optimisation to transform the robot configuration into actuation.

The CartStemContact and Trunk handling a cup examples show that optimisation methods cannot solve tasks that require the reconfiguration of contacts. Reinforcement learning algorithms are capable of solving these tasks but only after a long learning period. Each contact configuration leads to a given workspace, and in general, solving complex tasks involves reconfiguring contacts. Section 4.2 describes a soft robot by splitting its observation space into a set of configuration spaces. This allows the incorporation of prior knowledge into the learning process and the coupling of learning and convex optimisation.

4.1 Describe soft robots in configuration space with a proxy

As mentioned in related work (Chapter 2), various methods have been employed for high-level control in soft robotics. However, these methods either lack genericity or are cumbersome to compute because of the model they use, such as geometric models whose application is limited to slender soft robots. Extending these methods to more complex geometries and contact configurations is challenging. FEM is a good candidate for the simulation of complex soft robots with non-regular geometry, soft actuators, and actuation redundancy. However, the presence of many contacts and constraints in the environments in high level tasks can lead to demanding computation times. As a consequence, there is no general and efficient motion planning method for complex soft robots performing these tasks.

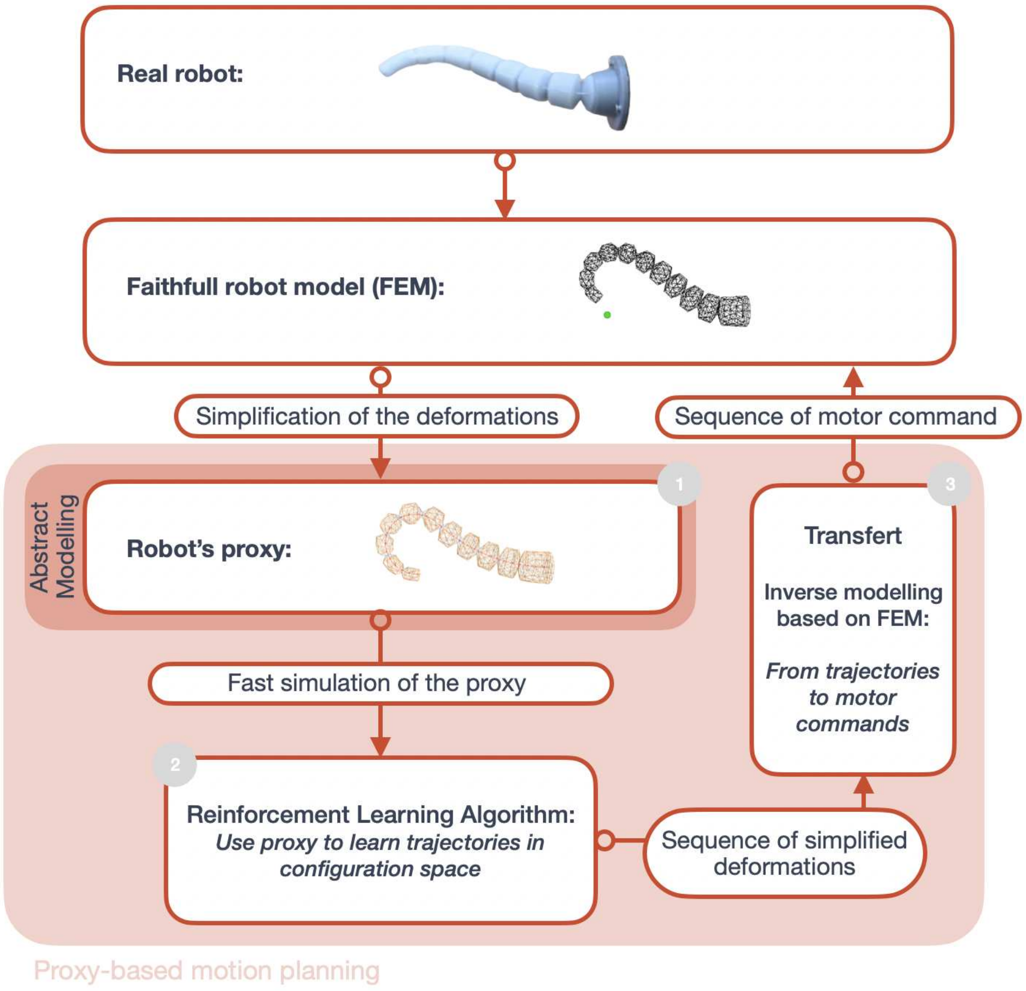

Real robot:
Faithfull robot model (FEM):
Simplification of the deformations
Sequence of motor command
Abstract Modelling
Robot's proxy:
1
3
Transfert
Inverse modelling based on FEM:
From trajectories to motor commands
Fast simulation of the proxy
2
Reinforcement Learning Algorithm:
Use proxy to learn trajectories in configuration space
Sequence of simplified deformations
Proxy-based motion planning

Fig. 4.1: Pipeline for using the proxy model to solve a given task in the case of the trunk like robot. Starting from a complete model, an abstract model called proxy is created (1). The proxy is then used to perform the learning and find a realistic trajectory of the robot (2). The actual actuation of the robot is calculated from the learned trajectory using an inverse model (3). The obtained motor control is applied to the complete model of the robot.

We assume that FEM provides an accurate model of the robot and we aim to find a computationally efficient motion-planning algorithm. For humanoid robots (see Section 2.4.2), the motion-planning problem is divided into two parts: generating a feasible trajectory in the configuration space and transferring it to the actuation space. This is done using simplified models of the robots, known as proxy models, which allow for the simulation of a simplified dynamic model at each time step during the motion-planning phase. This trajectory is then used as a reference to control the movement of the complete robot in simulation, resulting in the generation of the corresponding trajectory in the actuation space and addressing potential actuation redundancy.

The idea is to apply a similar method to soft robots by solving a part of the motion generation problem with a simplified model. The method is composed of 3 steps, as illustrated in Figure 4.1. The first step is to create a proxy model of the robot, that is, to extract from the faithful model a model that is less complex to simulate. This model is based on the description of the robot in a reduced configuration space. In the second step, the proxy model is used to generate a trajectory in this space using a learning approach. In the third step, the successive positions of the generated trajectory are used as a reference for a position controller based on the inverse FEM model. We will see that in addition to being a frugal method, that is, requiring less resources and computation time, the proxy model facilitates learning with a smaller or equal action space dimension than the number of actuators and an observation space directly defined by the model.

4.1.1 Method overview and setup

The proxy of a humanoid robot is typically constructed to produce a feasible trajectory in the task space, that is, a trajectory that can be reached by the complete system. This requires knowledge of the robot's workspace, which evaluation has been investigated for several decades in rigid robotics based on kinematic models. The evaluation and characterization of a soft robot's workspace can be far more difficult because of its dependency on the interaction forces and the material properties of the deformable body [Van+20]. It is necessary to use a mechanical model, which is often complex [AZK22; AZK21]. In addition, the potential actuation redundancy will lead to a multiplicity of configurations reaching the same task positions, which are challenging to identify. Therefore, we propose a motion planning method in which we define *feasible trajectories* as trajectories that can be approached with some errors. Moreover, these trajectories are primarily generated in the configuration space of the robot. The method comprises the following three steps.

- Step 1: Modelling. The proxy is constructed using computationally efficient mechanical elements to approximate the deformations of the soft robot and its interactions with the environment. This proxy represents the soft robot using a reduced state. In this work, we have described the state as a deformable skeleton.

- Step 2: Trajectory generation. The proxy is used to generate feasible trajectories in the configuration space using RL. Although we do not ensure that the trajectory can be perfectly followed by the robot, methods can be used to reduce the gap.

- Step 3: Transfer. The planned trajectory is used as a reference for an optimisation-based inverse control. This control is applied on an accurate FEM model and provides actuation inputs that can be reached by the soft robot. This step is used to transfer the trajectory from the configuration space to the joint space of the robot.

Each step is described in detail below.

Step 1: Modelling

Some soft robots are made of elements whose behavior can be described by the bending of a neutral fiber. This is the case of locomotive legs, fish robots, and trunk-type serial manipulators [MC16]. The main deformation mode of these robots is bending, so we use a bending parameter of the robot's neutral fibre as the state space. It is assumed that these bending elements can be identified. We can then base the construction of the proxy model on the theory of beams. Furthermore, we assume that the actuation imposes a constant pre-curvature on the beams. This is the case of conventional actuators such as cable actuators with a cable pathway parallel to the beam centerline, or pneumatic actuators when there is no contact [WJ10]. We also assume the beams to be inextensible to simplify the model further.

The proxy is created in three steps. In the first step, we identify the subparts of the soft robot that have similar behaviors in terms of bending and rotation/translation. These subparts can be identified by examining the location and nature of the actuators. In particular, soft pneumatic actuator with a cavity to be inflated can be represented by a beam with only one actuated bending. A slender elastic body deformed by pulling on at least three cables equi-distributed on its periphery can be modeled using a beam with two degrees of bending actuation. This reduces the state and isolates part of the complexity. In the cable space, we have a coupling between the cables and three actuation variables. Indeed, the cable tension space is defined with three actuation variables which must be constrained to ensure that the cables are pulled and not pushed. In the bending space, we no longer have this constraint, and only two actuation variables. In the second step, we extract the neutral fiber of the identified subparts to create a set of bending beams, called the skeleton of the soft robot. For the proxy, we chose to represent these beams with an Euler-Bernouilli model [SGW15]

$$P_{i,t} \sim f(L_b, S_b, M_b, \alpha_{b,t}, P^0_{b,t}, Q^0_{b,t}), b \in \mathcal{B} \tag{4.1}$$

where the points $P_{i,t}$ are uniformly distributed along the beams $\mathcal{B}$ and depend on $\alpha_{b,t}$ the pre-curvature of the beam b, $P^0_{b,t}$ the position and $Q^0_{b,t}$ the orientation of the base of the beam, L_b the length, S_b the stiffness, M_b the uniform mass. This equation includes design-related parameters (L_b, S_b, M_b) and actuation-related parameters ($\alpha_{b,t}$, $P^0_{b,t}$, $Q^0_{b,t}$). The value of the precurvature takes into account the actuation forces applied to the proxy. The actual curvature is given by the value of the pre-curvature and the external forces such as contacts that can deform the beam. The values of $P^0_{b,t}$ and $Q^0_{b,t}$ are either passive or actuated. Points $P_{i,t}$ represent the configuration of the soft robot. Considering beams' positions and not beams' curvature for the robot's state allows to model more efficiently the interactions with the environment such as contacts, amongst other advantages [MGK22]. To avoid numerical singularity, the beams are constructed such that their extremities either connect to another beam, connect to the environment (like the tip of a locomotor leg), or are controlled (like the tip of a manipulator). In the other cases, additional non-actuated beams are added, always remaining consistent with the geometry of the robot. All these beams allow to take into account the compliance of the robot structure. Finally, in the third step, a surfacic mesh representing the external surface of the soft robot is added on top of the assembled skeleton. This mesh is mapped onto the robot skeleton through skinning mapping [Dur+18], which creates a link between the surface contact forces and the resultant forces distributed on the beams. It is used to take into account the interactions between the robot and its environment.

The proxy creation requires the choice of parameters (maximum curvature α_b^{max}, mass M_b, stiffness S_b). We use a Bayesian approach [Sha+16] with a few iterations to optimize them. To obtain quick realistic results, we restrict the search space around hand-found values that provide realistic behavior in terms of deformation. As the proxy model is a simplified model, it is not useful to obtain the "best" parameters to describe it at the risk of spending a lot of time for minimal transfer results. The cost function used for Bayesian optimisation is given by the L_2 distance between points mapped on the faithful and proxy models. Both models follow an identical trajectory in terms of deformation and must exhibit similar behaviors on this trajectory. The trajectory is separated into optimisation and validation positions, which enables to check that the proxy is interpolating, that is, the parameters found for the optimisation positions are also good for the validation positions.

We have a lot of freedom to create the proxy. Other mechanical elements could be used for approximating the robot behavior, as discussed in Section 4.1.4, and different construction methods for the skeleton could be envisioned. The idea here is to demonstrate that we can use a very simplified proxy of the robot for control but not to demonstrate that our choice of proxy model is optimal.

Step 2: Trajectory generation

Step 1 provides a proxy that represents the robot in a reduced configuration space. Using this model and thus the definition of the robot in this space, a configuration sequence can be found to solve a given task. We use a RL algorithm to learn this configuration sequence, called the strategy. RL allows to solve high-level tasks and to adapt in the case of perturbations during these tasks. The trajectory is not generated in one single block but sequentially, one simulation step after the other. In the configuration space, the actions are the global deformations represented by the beams' pre-curvature $\alpha_{b,t}$ in Eq. 4.1, and the eventual translations $P^0_{b,t}$ and rotations $Q^0_{b,t}$ of the beams extremities. As introduced before, the points $P_{i,t}$ along the beams represent the robot configuration space and are chosen as the RL state of the proxy.

RL algorithms tend to exploit the weaknesses of the underlying model to determine the optimal strategy. Because our proxy model is an approximate model, there is no guarantee that the strategy found can be exactly realized by the real robot (in terms of deformation and contact with the environment), although the proxy parameters have been optimized to match the faithful model. To overcome this problem in the learning step, it is possible to introduce information about the a priori behavior of the robot, such as actuation symmetry or limitations of the actuation amplitude. This can be achieved by reducing the search space of the learned strategy and depends on the task at hand, as illustrated in the following.

Step 3: Transfer

Step 2 provides a configuration trajectory of the proxy that solves the desired task. The last step involves transferring this trajectory to the robot joint space using a high-fidelity FEM model. We implement a closed-loop position control based on the optimisation-based inverse model (see Section 2.2.3) and follow the feasible trajectory in simulation. In addition, two other control loops are used (see Figure 4.2). The first one corresponds to an external control loop linking the RL controller to the complete robot model, as shown in Figure 4.2.1. Due to friction contacts,

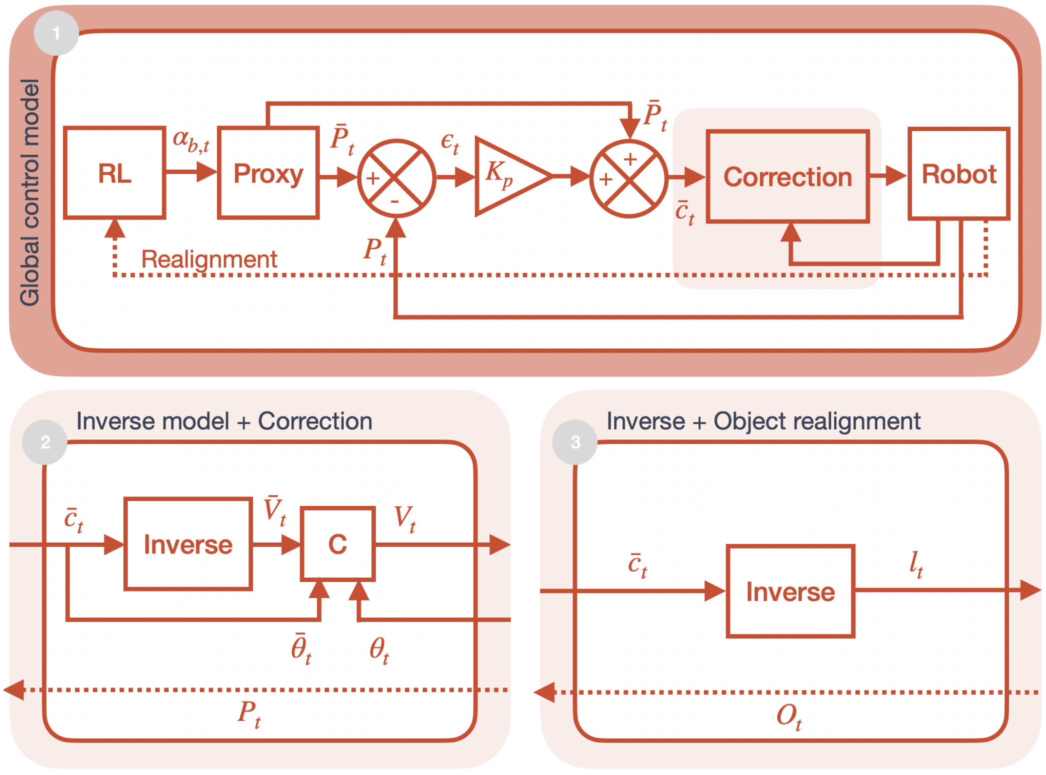


Fig. 4.2: Control scheme used during the transfer. (1) General case. A trained RL algorithm gives a curvature instruction $\alpha_{b,t}$ to the proxy model to solve a task. Following this instruction, a position reference $\bar{P}_t$ is extracted and compared to the current position of the robot P_t to form a corrected position command $\bar{c}_t$. The realignment allows taking into account the position actually reached by the robot P_t or the manipulated objects O_t. An inverse model gives an actuation command that can be corrected by taking into account information from both the proxy model and the robot. (2) The inverse model is used to find a pneumatic actuation V_t. A corrector C is used to correct the actuation to take into account the angles observed in the proxy $\bar{\theta}_t$. (3) Only the inverse model is used to find the cable lengths l_t.

some accumulation of errors can lead to a difference in behavior between the complete model and the proxy model, and a step of realignment of the proxy model is necessary. To do this, the relevant positions in the complete model are retrieved and imposed on the proxy robot model. It enables to obtain a proxy instruction that adapts to what happens with the complete robot model. For example, the position of the object manipulated with the complete model is used in the proxy model to obtain a sequence of action that depends of the real position of the object. The second control loop partially compensates for certain local differences that may exist between the proxy and complete models by correcting the actuation provided by the inverse model. For example, in the case of locomotion, these differences are due to different contact models. A closed loop control is used to adapt the actuation to force the two models to have similar contact behaviour. Figure 4.2.2 and 4.2.3 represent the correction modules for locomotion and manipulation respectively.

4.1.2 Experiments

Proxy modelling of Multigait and Trunk robots

To illustrate the method, we consider the Multigait robot for a locomotion task (see Section 3.1.2) and the Trunk robot for a manipulation task (see Section 3.1.2). In addition, the environment of the Trunk robot is composed of a rigid cube that can glide on a rigid planar surface, and the base of the trunk can translate in one direction. Following the steps described in Section 4.1.1, the two robots' proxies are shown in Figure 4.3.

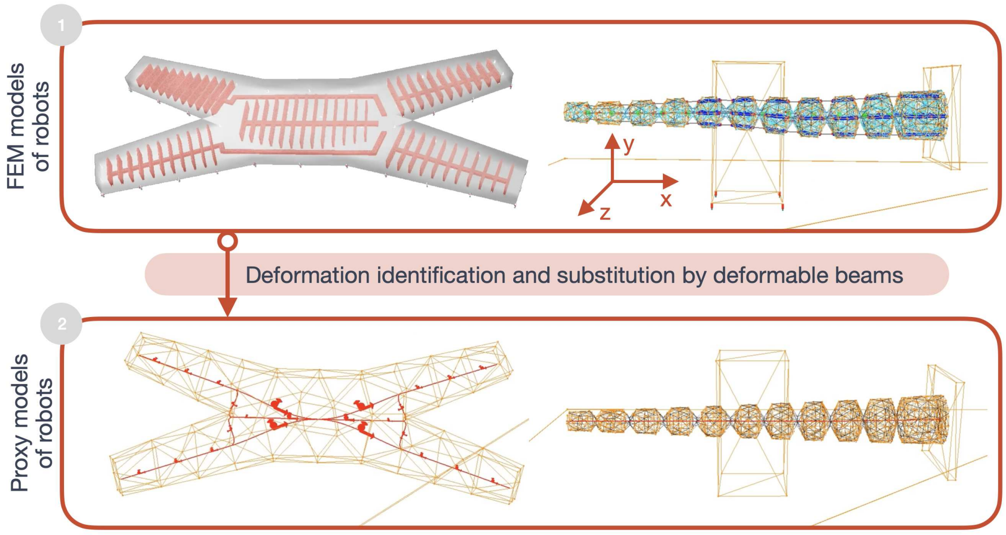

1
FEM models of robots
y
x
z
Deformation identification and substitution by deformable beams
2
Proxy models of robots

Fig. 4.3: Representation of the FEM (1) and proxy models (2) of the Multigait and Trunk soft robots. The deformable beams composing the skeleton of the proxy model are depicted in red. The external meshes for both models are identical.

The Multigait robot is a robot with five cavities, each of which represents one bending degree of freedom. Therefore, for the proxy, we consider the robot skeleton and replace each cavity with a beam with one bending degree of freedom. Additional beams are added to the skeleton following the edges of the robot in order to satisfy the beam coupling condition introduced in Section 4.1.1. We obtain a simplified model with 19 nodes (4 per leg and 3 for the central part). The Trunk proxy can be seen as a skeleton made of two beams placed end-to-end, each with two degrees of freedom of bending. Thus, the Trunk is separated into two parts, and each part is represented by a beam. The mobile base of the robot, which can translate along the x-axis, is the same for the proxy and complete models. We obtain a simplified model with 30 nodes.

Solving the task using RL algorithm on the proxy model

For the generation of the trajectory with RL, we use SofaGym for the implementation of the environment and the PP0 algorithm with the following characteristics. We emphasize that the contribution of this study is independent of the selected RL algorithm.

For the Multigait robot, the state of the robot is defined by 20 points uniformly distributed on its skeleton. We limit the action space by considering the all-or-nothing positive bending of the actuators while imposing symmetric activation to ensure that the robot moves forwards. The reward is defined in Section 3.1.2 as the position increment at each timestep. In this environment, one iteration in SofaGym corresponds to 0.6s of simulation. For the Trunk robot, the state is defined by 16 points uniformly distributed on its skeleton, the position of the center of the cube, and the position of the goal. The actions are defined by two possible bendings of the two beams and the translation of its base along its main axis. The main reward is given by the distance between the cube and the goal. We also added a negative weighted sum of the distance between the trunk and the cube to favor contact. In this environment, one iteration in SofaGym corresponds to 0.3s of simulation. The task is solved when the cube is less than 8 mm from the goal, which corresponds to 5% of the Trunk length.

4.1.3 Results

Once the proxies are defined, we have to perform steps 2 and 3 of the method, namely the trajectory generation and the transfer to the complete model. In this Section, we present the results obtained in terms of simulation time, parameter optimisation, learning, and transfer.

Improved simulation time and learning time

The simulation times are evaluated in terms of RTF. All tests are performed under the same calculation conditions. The proxy and FEM models are placed in a configuration where the number of contacts is maximal. The results are summarized in Table 4.1. The proxy is 250 times faster for the Multigait and 3 times faster for the Trunk compared to the FEM model. The RTF gain is greater for the Multigait robot, because the Multigait's FEM mesh (34k nodes) is much larger than the Trunk's FEM mesh (900 nodes). Similarly, in the context of this study, the proxy models can have RTF > 1 that corresponds to a computation time compatible with real time, which is advantageous in a learning framework. In fact, the calculation is as fast as retrieving data for live learning on a real robot. This method is frugal in the sense that it is less computationally expensive than the use of the complete models. In fact, it is possible to solve slow models in parallel, which reduces the computation time, but increases the computational burden. Simplified models reduce the computational burden, and therefore, the computation time.

Tab. 4.1: RTF obtained for the two robots, Mu. = Multigait, Tr. = Trunk, F. = FEM, P. = Proxy, MOR = Model Order Reduction

Δt_{simu}	F. Mu.	MOR Mu.	P. Mu.	F. Tr.	P. Tr.
0.01	0.002	0.06	0.57	0.22	0.62
0.02	0.004	0.12	1.06	0.49	1.33
0.03	0.006	0.17	1.51	0.72	1.96

Proxies parameters optimisation

The actions used to generate the parameter optimisation trajectory are those that bring the robot into extreme bending and resting configurations. We fix the time step to 0.01s, and we split the trajectories in optimisation and validation sets. For the Multigait robot, we consider a 10.8s trajectory composed of all-or-nothing actions. We use the position of the bottom face in the two models for the optimisation. For the Trunk robot, axial symmetry is used. We consider a 0.6s trajectory composed of 2 actions. The positions of the external mesh points are used for optimisation. As the optimisation was performed without interaction between the cube and the robot, we had to optimize the cube weight a posteriori to obtain the same behavior during interaction. This can be improved in future work. To ease the transfer, a safety parameter is used to force the proxy to find underestimated actuation magnitudes compared with the actuation limits of the actual system. This allows for compensating the deformation error between the models.

For the Multigait robot the average error defined in 4.1.1 on the optimisation positions is 8.02 mm (4% of the characteristic size of the robot), and 10.98 mm for the intermediate positions (5.5% of the characteristic size of the robot). The characteristic size of the robot corresponds to its length along the axis of motion, that is along the x axis. For the Trunk robot, we obtain an average error of 10.2 mm for the positions used for optimisation and 9.85 mm for the intermediate positions. This error corresponds to 5.2% of the characteristic length of the robot.

Solve the task with the proxy and transfer to the complete model

For both robots, the PPO algorithm determines a strategy on the proxy that solves the task. The number of simulation steps for learning to control the Multigait is approximately 20000, and 200000 for the Trunk. These strategies are illustrated in Figure 4.4. The strategy transfer to the complete model of the Multigait robot and the Trunk robot are illustrated in Figure 4.4.2 and 4.4.4 respectively. We consider the transfer to be successful when the models exhibit similar behaviors with respect to the cost function used for training.

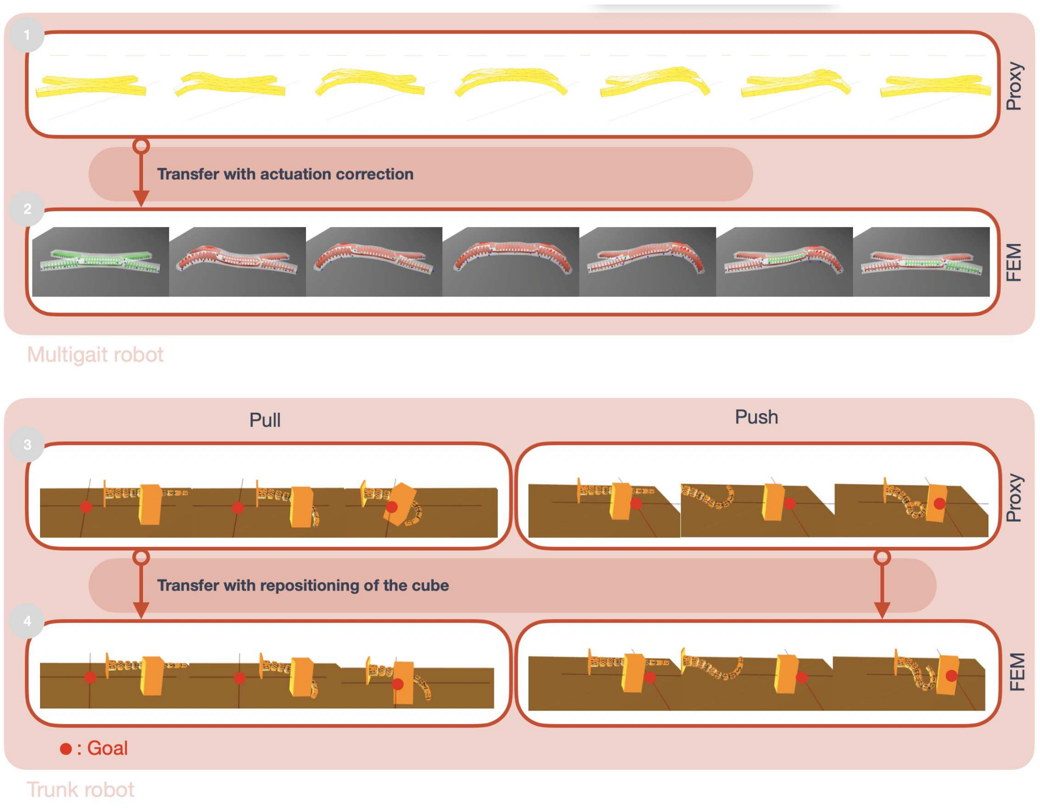

1
Proxy
Transfer with actuation correction
2
FEM
Multigait robot
Pull
Push
3
Proxy
Transfer with repositioning of the cube
4
FEM
● : Goal
Trunk robot

Fig. 4.4: Example of strategy obtained on the proxy model and transferred to the complete model in the case of the Multigait and Trunk robots. (1) Strategy learned with the proxy model of the Multigait robot. (2) Transfer of the strategy to the FEM model of the Multigait robot. (3) Strategy learned with the proxy model of the Trunk robot. (4) Transfer of the strategy to the FEM model of the Trunk robot. Despite differences in deformation, the strategy can be transferred between these two models.

For the Multigait robot, we use the actuation correction shown in Figure 4.2.2. The inverse model is used to calculate the actuation according to the position reference given by the proxy model. As the inverse model does not consider the angle of incidence of the contact points, a correction step allows to follow both the position and orientation of the contact points. The realignment steps are performed with a fixed time step. After the transfer of the strategy from the proxy model to the complete model, the complete model moves by about 5.83 mm against 5.53 mm for the proxy model, which corresponds to a difference of 5%.

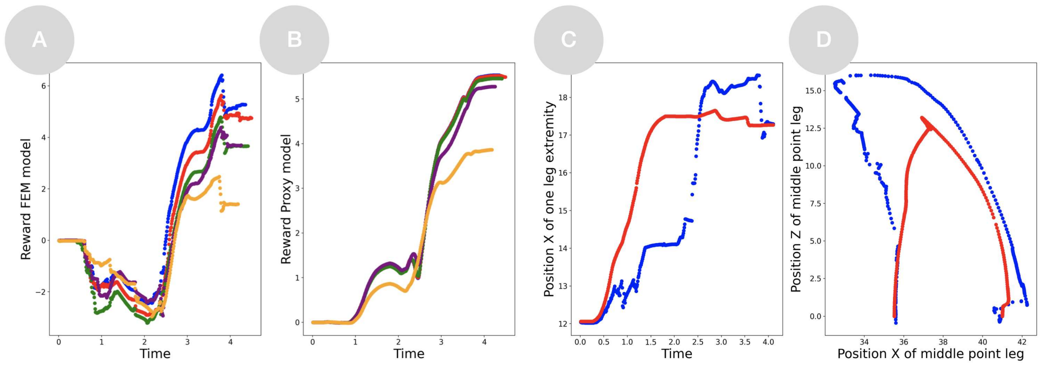


Fig. 4.5: Results for the Multigait robot. A) Reward obtained with the FEM models (mm) by following different proxies over the time (s). B) Results obtained with the corresponding proxies over the time (s). For A) and B), the blue line represent the results obtained for the optimized proxy, while other colors correspond to proxy with noisy parameters. C) Evolution of the position of one leg's end in contact with the ground, along the x-axis (mm). D) Evolution of the position of one leg's middle point in the plan (x, z) (mm). For C) and D), the blue dots correspond to the position obtained with the FEM model following the position of the proxy model given by the red dots.

Figure 4.5 illustrates the results for the Multigait robot. Concerning the reward (Figure 4.5.A), the FEM model starts with a negative reward and exceeds 6mm before dropping back down. This behavior does not exist in the proxy model. This is because contact conditions, and in particular friction and contact flexibility, are not identical between the two models. However, the significant increase in reward occurs at exactly the same time (2 seconds). The two models behave identically thanks to the control loops, despite the few differences that may exist between them. Concerning the position of the point at the end of one of the Multigait's legs in contact with the ground (Figure 4.5.C), the FEM model is stuck at the beginning. The inverse model seeks to minimise an average error over all the points defining the reduced configuration, and not over particular contact points. The actuation is corrected to reach both the targeted contact point angle and position. Concerning the position of the point at the middle of one of the Multigait's legs (Figure 4.5.C), we can see that the the FEM model deforms much more than the proxy model, in order to meet both the positioning and angle requirements. In the case of the Multigait, the proxy should give a general idea of the robot's configuration and provide realistic instructions for the angle and position of contact points.

We want to evaluate the robustness of the method with respect to the precision of the proxy. To do this, we construct degraded proxies. The proxy used for generating the trajectory in Figure 4.4 is considered as the reference. Each degraded proxy is obtained by adding uniform noise to the parameters of the reference proxy, ranged from 0.001 to 0.1 of the value of the parameters. The strategy is learned with the degraded proxy and the transfer is performed with the FEM model. Figure 4.5.A and 4.5.B show the reward obtained with the different proxies. We can see that in each case our planning method leads to a trajectory that increases the reward, demonstrating a certain robustness to the proxy accuracy. Similar results are obtained when the parameters are close to those of the reference proxy. However, poorly chosen parameters can lead to degraded results, whether in terms of the reward of the proxy model or transfer.

For the Trunk robot, we study two cases: the positioning of the cube along the x-axis and the positioning of the cube at a position (x, z). The used controller is illustrated in Figure 4.2.3. The registration step is used to take into account the position of the cube: after several actions, the position of the cube detected in the complete model is imposed on the cube of the proxy model. Therefore, if the cube is not in the expected place, there is a re-planning step where RL-based motion planning adapts the actuation of the trunk to ensure the success of the manipulation task. For positioning along the x-axis, the relative positioning error of the cube $\frac{|r_t^{proxy} - r_t^{FEM}|}{|r_t^{FEM}|}$, where r_t corresponds to the reward, is approximately 3% between the two models. We obtain the same results when the goal is in position (x, z), even if the position is outside the learning space. In the latter case, the final position of the cube corresponds to an error of 7% compared with the position given by the proxy model.

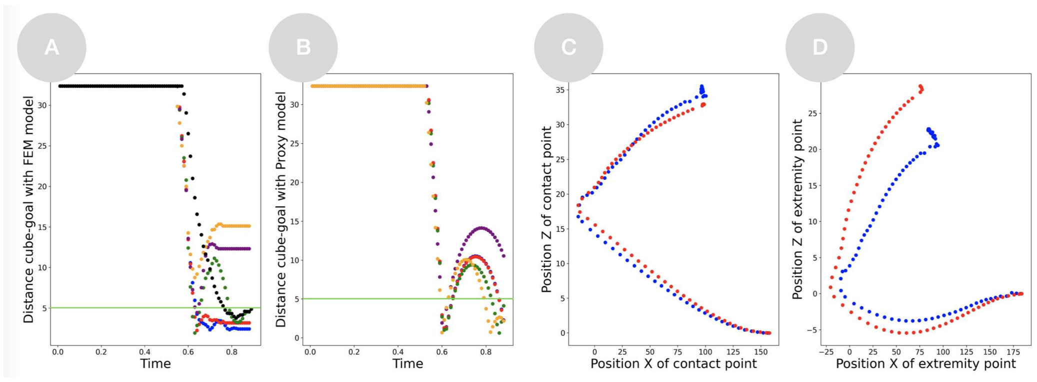


Fig. 4.6: Results for the Trunk robot. A) Cube to goal distance obtained with the FEM models (mm) by following different proxies over the time (s). B) Cube to goal distance obtained with the corresponding proxies over the time (s). For A) and B), the blue line represent the results obtained for the optimized proxy, while other colors correspond to proxy with noisy parameters. The algorithm stops when the distance between the cube and the goal is less than 5mm, represented by a horizontal green line. The task is then considered solved. C) Evolution of the position of the point in contact with the cube in the plan (x, z) (mm). D) Evolution of the position of the tip of the Trunk in the plan (x, z) (mm). For C) and D), the blue dots correspond to the position obtained with the FEM model following the position of the proxy model given by the red dots.

Figure 4.6 illustrates the results for the Trunk robot. The two models are close in terms of positioning at the point of contact between the cube and the Trunk.

However, even close at the begining of the task, the behavior is different for the tip of the Trunk as deformations are not modeled in the same way. This does not impact cube handling because the positioning of this point has little influence on the task to be performed. In addition, there is a bell in the distance measured using the proxy model and not in the FEM model. This is due to the fact that the friction is not the same in the two models. Distance minimization occurs simultaneously; therefore, both models push the cube towards the goal at the same time. The global behavior is the same.

We also compare the previous performances obtained with the reference proxy to those obtained when we use other parameters for the proxy or when we directly use the FEM model of the Trunk with a RL algorithm. For the different proxies, the results are the same as the results obtained for the Multigait robot. Learning stops when the distance between the cube and the goal is less than 5 mm, and both the FEM model with RL and the proxy method manage to solve the task. It seems that, in this particular case of pushing the cube, the proxy leads to better results (2 mm between the cube and the goal for the proxy method, against 4.5 mm for the FEM model with RL). In any case, the task is solved using a proxy model with a highly reduced computation time, which shows its interest.

4.1.4 Discussion

The previous results show that the method allows to control two robots: one in locomotion and one in manipulation tasks. Here, we discuss the different steps involved in this method.

Discussion on the proxy modelling step

In this study, we chose to build the proxies using the robot's skeleton. The deformations of this skeleton generate movement, and are caused by actuations and interactions with the environment. This allows having a minimum set of geometric parameters for the motion planning, in particular in the case of redundant actuation (as in the Trunk case). Focusing on the deformations allows to decrease the dimension of the action space during the learning phase and move the management of antagonistic actuation to the transfer step. Consequently, the temporal performances of the simulations and the motion planning in general are improved.

Although we have considered inextensible beams, considering extension and even torsion is theoretically possible, for example, by using Cosserat beams to build the skeleton. In practice, we did not find any problem for controlling the Trunk, but this question must be asked in other cases with different interactions with the environment or with a different robot. Finally, describing the robot's state by representing its skeleton using deformable beams is one of the many approximations that can be used to build the proxy model. It could be possible to use other approximations, such as mass-springs, articulated bodies, or particles. We can also use Model Order Reduction (see Section 3.1.2). The complete method would remain the same, performing training on a simplified model and then transferring it to a complete model. To be useful, the simplified model must provide a good trade-off between computation time, reduced motion parameters and realistic behaviour.

Discussion on the trajectory generation step

The focus of this work was not to develop sophisticated RL strategies. The strategies found are for simple tasks, and we were satisfied with trajectories that solve the task. Because we know that the proxy model is approximate, the best solution found in the configuration space is not necessarily the best one for the FEM model. However, the transfer step allows the FEM model to use the strategy found with the proxy to solve the given task. In this study, we focused on the general method and transfer rather than on obtaining the optimal strategy with the proxy for a wide variety of tasks. For the Trunk robot, using RL on several goal positions allows the adaptation of the instruction after a realignment step. For the Multigait robot, we use the symmetry to constrain the strategy. The result is a less complicated strategy than that with non-symmetrical continuous actuation, but it is easier to transfer.

For the Multigait, we can use RL and the hypothesis on the strategy search space to understand the weaknesses of the underlying proxy model. When the symmetry assumption is relaxed in the learned strategy, the left front leg inflates while the right front leg deflates. With this relaxation, there is a difference of approximately 45% between the two models. When there is a strong asymmetry of actuation, there is a large difference in deformation between the models, which is exploited by the proxy and is not reproducible in the FEM model. When the all-or-nothing assumption is relaxed, that is, all combinations of symmetric continuous actions are used, there is a difference of approximately 80% between the two models. The calibration is incorrect: proxy parameters were found with a trajectory generated by all-or-nothing actions. Outside these trajectories, the behavior of the proxy can be very different from that of the real model. The hypotheses on the strategy have to correspond with the sequence on which the proxy parameters have been optimized. Potential ideas to overcome this problem are to perform a proxy-trajectory co-optimisation, to add penalty terms in the reward when the proxy behavior deviates from the real robot behavior, or to use other sim-to-real methods to improve the transfer between the proxy model and the complete model.

Discussion on the transfer step

One of the key features of this method is the simulation of the transfer. Compared to the methods that make a direct link between the simplified model (particle, voxel, etc.) and the physical prototype, we have an intermediate step. It is possible to test and correct the learned strategy using the FEM model before transferring it to a physical prototype and to determine whether the strategy is viable. Generating the actuation trajectory with the inverse method on the full FEM model allows correction of the proxy errors and increases the chances of successful transfer. FEM simulations rely on models that have been already validated on physical prototype in numerious publications, including a wide variety of soft robotics systems such as manipulators, grippers, and mobile robots. Closed loop strategies can be used on a real robot to accurately follow trajectories. We are convinced that if we manage to transfer the learned policies to a FEM model, there is a good chance that they will be transferred to a physical prototype. The most important point is, therefore, the transfer to the FEM model, and the transfer to a physical prototype is another topic that will be investigated in the future.

For the Multigait robot, what is important for the transfer is that the position and the angle of the contact points of the proxy during locomotion are realistic. However, the actuation of the proxy is underestimated, and by simply following the trajectory provided by the proxy model, the model cannot follow both these contact positions and angles. Correcting actuation with respect to this angle modifies the pressure in the cavities under these two conditions. The particularity of the Trunk robot is that it interacts with an object. Due to the differences between the FEM and proxy models, the behavior during contact is different (for example, the complete model collapses more than the proxy model). When the goal is in a position (x, y) outside the learning space, the accumulation of errors causes, at some point, the FEM model to not touch the cube while the proxy model does. However, the realignment step occurs, and the instruction is adapted to consider the new position of the cube.

Even if the behaviors are different between the models, the interactions are located at the same places and have the same characteristics, which enables the use of a proxy model to control the complete model. Currently, the transfer method imposes a 1 to 1 equivalence between the proxy and complete model positions. We can get closer to what is done in humanoid robotics for locomotion, for example, by using a proxy to find a contact sequence and then follow it.

4.2 Describe soft robots using a set of configuration spaces

As shown in the CartStemContact example, optimization-based inverse modelling can be used to control soft robots, but these approaches have intrinsic limitations especially regarding contact reconfiguration. In rigid robotics, various control methods are employed to handle contact, in planning and predictive control [Pan+23; Mar+22; Mar+23; UA22]. However, soft robots control based on FEM is limited to one time step with optimisation and inverse modelling (see section 2.2.3), except for simplified robot models [SK22; LSH19]. Although this method can handle contact, it does not account for contact reconfiguration as breaks in the optimisation space occur. In another hand, can be used to solve complex tasks that cannot be solved using optimisation methods thanks to the 0th order approximation of the gradient and to their exploratory aspect [Lid+22]. However, this method is generally less interpretable or less sample-efficient than model-based optimization approach. The goal of this section is to present a new approach for incorporating prior knowledge into the learning process, which is represented as a graph of configuration spaces. This graph divides the robot states into distinct hand-defined configuration spaces, and sequences of actions are used to navigate between them. It enables the combination of learning and convex optimisation, enhances interpretability and reusability, and enables the use of expert trajectories in the learning process.

4.2.1 Configuration spaces graph, agents and learning

We propose a modular method that combines the advantages of learning and prior knowledge (optimisation-based or user-supplied data) in control problems. It is formalized as an HRL framework based on a user-defined knowledge graph, where the nodes represent configuration spaces of the robot. A configuration space corresponds here to a set of robot configurations that share the same contact configuration.This method employs multiple agents to identify the current node in the graph (e.g., which contact configuration the robot is in), navigate through the graph (e.g., how to change the contact configuration), and solve the task in each node of the graph. This structure allows for the integration of various optimisation methods. It enables also to reuse pre-trained agents and provides interpretability. The details of this approach are presented in the following sections.

Overview of the method

The user defines a graph of N configuration spaces S_i covering the state space of the robot S (see "selector" in the next subsection for details). The use case limits the state space, and the covering $\sqcup_{i\leq N} S_i = S$ is defined according to the task. Each configuration space is identified with a vector $h_i \in \mathbb{R}^{d_s}$. The graph naturally induces a notion of neighborhood V_i of the configuration space S_i. Two nodes are linked when it is possible to go from one configuration space to another by following a sequence of actions without going through a third different configuration space.

Given this graph, several agents are used to solve the task:

1. The Selector $\mathcal{S}$ determines from the state s_t in which configuration space the robot is. To do this, it calculates the probability that the current state belongs to a given configuration space:

   $$\mathcal{S}(S_i|s_t) = p(s_t \in S_i) \tag{4.2}$$

   where $S_i|s_t$ means "being in configuration S_i knowing that the robot state is s_t". The Selector is deterministic when there is no uncertainty about the configuration spaces. Indeed, when the Selector is learned, there is uncertainty and the Selector's aim is to maximise the probability $p(s_t \in S_i)$. When the Selector is implemented using collision detection algorithms, there is only one possible choice and the probability is 1 or 0. The Selector is the agent that implicitly defines the knowledge graph. It associates each state with the configuration space that contains it, either by learning from data labeled according to the configuration spaces or by using an external criterion, such as a collision-detection algorithm.

2. The Evaluator $\mathcal{E}$ determines if the robot should stay in the current configuration space to solve the task or if it should change. It gives the probability of success p_{succ} that the task can be solved in the configuration spaces:

   $$\mathcal{E}(S_j|S_j \in V_i \cup \{S_i\}, s_t \in S_i) = p_{succ}(s_{t+1}|s_{t+1} \in S_j, s_t \in S_i) \tag{4.3}$$

   This probability corresponds to the action of the Evaluator (network output), and is trained to maximise its cumulative reward. Depending on the configuration space that maximises this probability, the Evaluator uses one of the two last node-specific agents to find the next action: the external agent or the internal agent.

3. An external agent $\mathcal{A}_{ext}$ is used to switch from the current space S_i to another $S_{j \neq i} \in V_i$ in the neighborhood. This agent uses options (see Section 2.2.2).

4. An internal agent $\mathcal{A}_{int}$ is used to solve the task inside the current configuration space. This agent uses primitive actions.

Each agent can be either learned or not, and can share information with other agents. The Selector and the External agents are independent of the task to be performed, while the Evaluator and Internal agent are not. An overview of the method and the link between agents is shown in Figure 4.7.

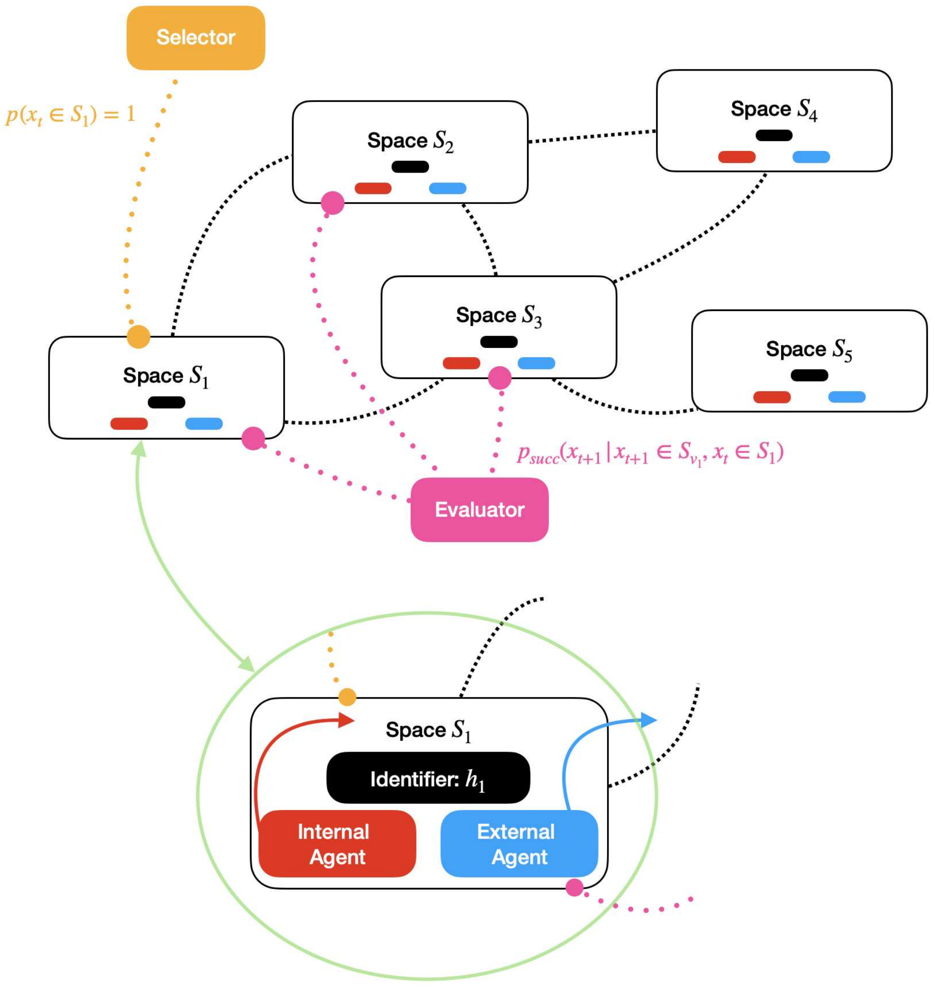


Fig. 4.7: Schematic of the different elements constituting the knowledge graph. Each node in the graph corresponds to a configuration space, and each observation belongs to one node. Each node has an identifier, a neighborhood (other configuration spaces), an internal agent (in red), and an external agent (in blue). The Evaluator (in pink) can either belong to a node or be shared between the nodes. The Selector (in orange) is global. Three steps are performed. (1) The Selector determines in which configuration space the robot is. (2) The Evaluator decides in which configuration space to go. (3) The internal agent or external agent solves the task in the current space or moves to a different space.

Structure and Training of the different agents

In this section, we present all the trained agents, using off-policy learning[1] and replay buffers[2].

The Selector: The configuration spaces S_i are labeled according to the prior knowledge. The Selector is based on an attention mechanism [Vas+17]:

$$\alpha = softmax(\frac{QK^T}{\sqrt{d_s}}) \tag{4.4}$$

where $K = F(s_t)$ is the encoded current state, F a Multi-Layer Perceptron (MLP) network, Q is the matrix of the identifiers h_i, and d_s the dimension of the identifiers. It gives the similarity between the encoded current state and each identifiers. The softmax function gives a probability $\alpha_i = p(s_t \in S_i)$ that the state s_t belongs to the space S_i. The current configuration space is given by the maximum of the probability. For the training, labeled data are collected and the Selector has to predict the correct label in a supervised way according to:

$$\forall s \in S_i : \alpha_i = 1 \text{ and } \alpha_{j \neq i} = 0 \tag{4.5}$$

The Evaluator: The Evaluator finds the probability of solving the task in $V_i \cup S_i$ using an attention mechanism as well. For the Evaluator, K is the matrix of the identifiers and Q is the encoded current state. The result is given by $\alpha \sim \mathcal{N}(\alpha_{mean}, \alpha_{std})$ where α_{mean} and α_{std} are evaluated using Eq. 4.4 and $\mathcal{N}$ is the normal distribution.

The rest of the learning is a classical RL learning process. Two different transitions are used, depending if the Evaluator chooses external or internal agents. They can be both express as a list:

$$(s_{t+k}, a_t, s_{t+T}, \sum_{t'=t+k}^{t+T} r_{t'}, \delta) \tag{4.6}$$

[1]Off-Policy learning: the strategy that is used for action selection is different from the target strategy that an agent is trying to learn.

[2]Replay buffer: where the algorithm store trajectories when it executes a strategy in an environment.

with k an integer in $[0, T-1]$, $s_{t+k} \in S_i$, $s_{t+T} \in S_i$ for internal agents and $s_{t+T} \in S_{j \neq i}$ for external agents, T the duration of the option with $T = 1$ for internal agents (primitive action) and $T \geq 1$ for external agents, and δ a flag to indicate the termination of the episode. Internal and external agents are used to explore the state space. The Evaluator can be penalized if it uses internal agents to move from one configuration space to another one.

The internal agents: They are trained with all the transitions starting from the current configuration space. The internal agents may not find solutions in this space and have to explore other spaces to do so. Although the Evaluator has to choose the external agents for this exploration, it may make mistakes or converge slower than the internal agents. To solve this problem, each internal agent has information about its neighborhood. The idea is to update the Q-function (see Equation 2.8) of each internal agent using the Q-function of its neighboring internal agents for the boundary states (which allow to switch from one configuration space to another in one action). This allows the internal agents to know if it is beneficial to visit adjacent configuration spaces or not.

The external agents: The implementation of external agents is not dependent on the task to be performed and is based on concepts from HER [And+17] as follows. The current state s_t is augmented by the value of the identifier of the target configuration space h_i to form the input of the external agent $[s_t|h_i]$ where $|$ represents the concatenation. The idea is then to say that a trajectory τ allowing to reach a configuration space S_i does not allow to reach $S_{j \neq i}$. $[s_t|h_i]$ following τ is rewarded while $[s_t|h_j]$ is penalized.

4.2.2 Robots and creation of configuration spaces

We illustrate our method with two examples: the CartStemContact (see Section 3.1.2), and the RodManipulator, a manipulator made of two-soft cylindrical robots called fingers (see Section 3.1.2). These robots and their associated configuration spaces graphs are shown in Figure 4.8. Both systems are simulated using SofaGym. The learning process is implemented using an open-source implementation of SAC [KG17; Par20]. Other RL algorithms were also used in SofaGym and led to comparable performance. The hyperparameters are the same between the baseline method and our method.

The CartStemContact example:

The CartStemContact can be considered as a simplified soft robot. As mentioned in Section 3, it highlights the control issues associated with contact configurations. It is possible to solve this task using RL algorithms; however, the learning time is long. As we know the solution of this example, we can compare to it.

An episode last at most 30 iterations and the reward is equal to the horizontal distance between the tip of the beam and the goal. The position of the contact and the position of the mobile base are randomly initialized. The state s_t is $[x_{cart}, x_{tips}, x_{left}, x_{right}, l_x, l_z, x_{goal}]$ and contains the horizontal position of the mobile base, of the tips, of the obstacles and of the goal, and the dimension of the obstacles. The mobile base is controlled in position with a constraint on the maximum speed of the cart.

The state space is split according to the contact between the beam and the obstacles, as defined by the geometric conditions:

$$x_{left} + 1/2\sqrt{l_x^2 + l_z^2} < x_{cart} < x_{right} - 1/2\sqrt{l_x^2 + l_z^2} \tag{4.7}$$

This partitioning enables the identification of areas that can be reached by the end effector of the robot. This process does not require an explicit expression of the physical model, but rather only the observation of the robot's current state.

The RodManipulator example:

The method is illustrated by an example of object manipulation. The RodManipulator robot is used to catch and manipulate an object using two soft fingers. Each finger can bend in four directions and translate vertically. The object to be manipulated is a rod with a square cross-section placed on a table, and the goal is to rotate it using both fingers to reach a target vertical orientation. The state of the robot is defined by the positions of three points at the end of each finger, the position and orientation of the center of the rod, and the target orientation. The reward is based on the normalized angle covered by the bar during its movement. Because there is no simple formulation for modelling the mechanical behavior of the robot, a collision detection algorithm is used to identify the contact configuration. In this case, the Selector is deterministic and computed rather than learned. Contact

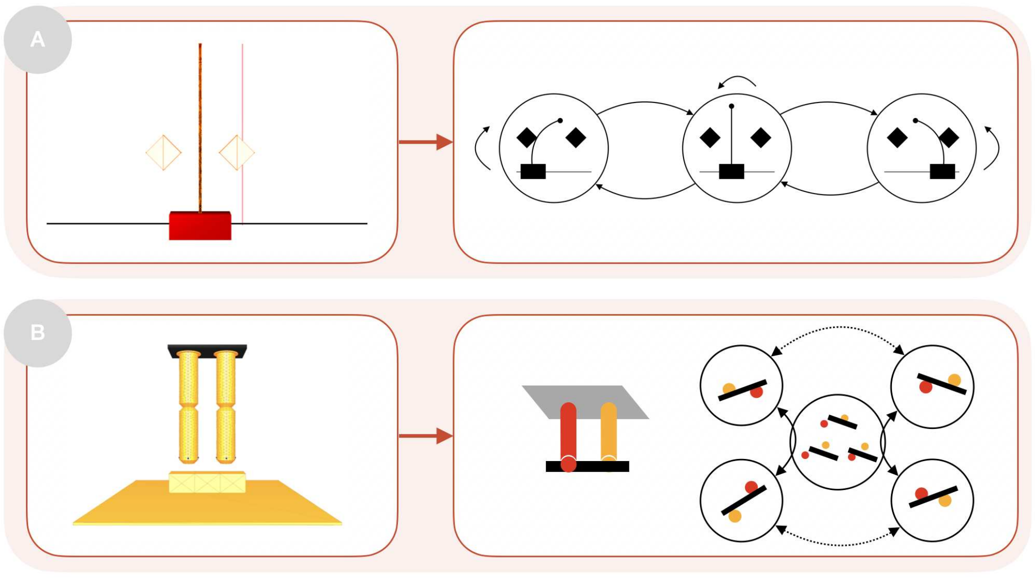


Fig. 4.8: Splitting the state space into different configuration spaces (right) for two soft systems (left) based on the contact configuration. (A) CartStemContact. (B) RodManipulator. In this example, some contact configurations are gathered in one configuration space, not useful for the manipulation task.

configurations that are not useful for the task are gathered in the same configuration space.

4.2.3 Results

Solving the CartStemContact and convex optimisation

The Selector's training is achieved by randomly collecting states and using them to train the model. In supervised learning, the data set is generally separated into a training set and a validation set. In this work, the validation set consists of 20% of the labeled data, and the model has a success rate of over 99.7% for 50000 test samples. The Selector manages to associate each observation to its corresponding configuration space.

The learning results are shown in Figure 4.9.A. The proposed method can solve the task in 80000 iterations, whereas the baseline SAC requires 140000 iterations. This corresponds to a 42% reduction in the number of iterations required to achieve the same performance level. Both approaches have the same limitations, due to the definition of the reward: when the goal is close to the bounds of the obstacle, the cumulative reward is lowered by getting stuck to the obstacle.

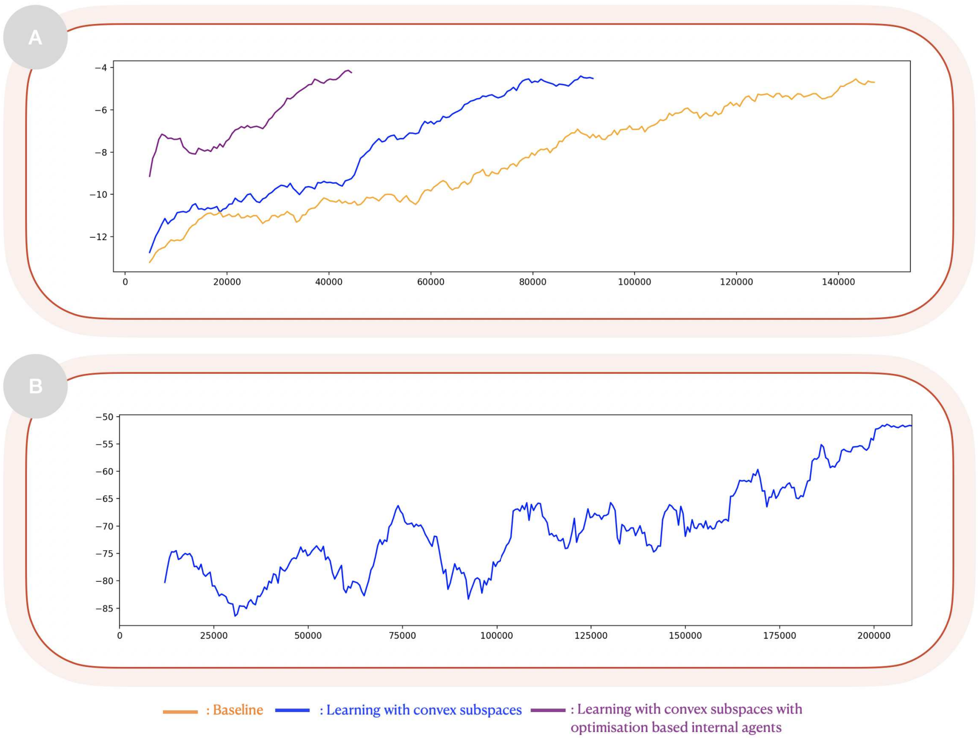


Fig. 4.9: Learning results, reward as a function of the iterations. (A) Results obtained with the SAC algorithm (orange), with our method (blue) and with our method with internal agents performed with convex optimisation (purple) in the case of the CartStemContact. The learning conditions are the same in all three examples. The initial difference comes from the fact that the first results are obtained after 500 iterations, and that the method with internal agent performed with optimisation learns to solve the task faster than the other methods. (B) Results obtained with our method in the case of RodManipulator. Sliding average is used to facilitate the reading of the results. The size of the windows for the sliding average is approximately 2.5% of the number of iterations.

The learned internal agents can be replaced by convex optimisation algorithms. The training for the other agents remains the same. In this setup, the task is solved in 40000 iterations, which represents a 72% reduction in the number of iterations compared to the baseline. To validate the reusability of the agents, the fully trained internal agents of the network are replaced with convex optimisation algorithms. When one or several internal agents change, the overall learning behavior remains the same, as the value of the cumulative reward changes by only 0.64% compared with all learned internal agents. These features make the approach particularly easy to combine with the existing methods.

Manipulation of the rod and expert trajectories

We define two expert trajectories. They allow the fingers to: 1) move away from the rod, one finger being brought forward and the other backward; 2) reverse the position of the fingers, the one that was forward now being backward and vice versa; 3) bring the fingers in contact with the rod in a new contact configuration. An illustration of these expert trajectories corresponds to the four images of the first row in Figure 4.10. The expert trajectories are hand-defined to guide the system from one contact configuration to another, but they do not specify the target configuration space. In fact, as they are not designed for a specific state, the Evaluator cannot use them to precisely reach a given configuration. However, the Evaluator can use these expert trajectories (external agents) to move out of the configuration space and modify the contact configuration. As these trajectories involve moving the fingers away from the rod, a penalty is included in the reward to encourage the Evaluator to use them only when this increases its long-term reward.

The cumulative reward for the RodManipulator is shown in Figure 4.9.B. The proposed method learns to use expert trajectories effectively to solve the task. An example of solving the task for a 280° target is shown in Figure 4.10. Starting without contact, the algorithm uses an external agent to create contact. The rod is then manipulated until it is impossible to move further without changing the contact configuration. Finally, an expert trajectory is used to reconfigure the contacts and solve the task.

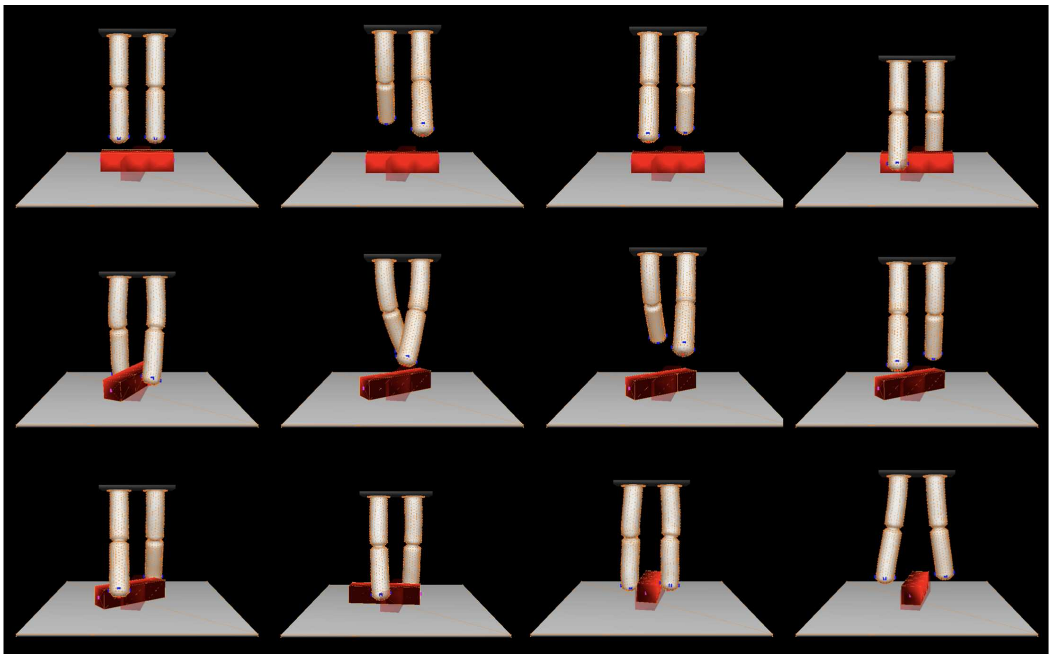


Fig. 4.10: Example of RodManipulator task resolution for a target angle of 280°. Starting without contact, the algorithm first uses an expert trajectory to position the robot in contact with the rod. The rod is then manipulated until it reaches a configuration where it is not possible to move further without changing the contact configuration. The Evaluator then uses another expert trajectory to reconfigure the contacts and continues to rotate the rod until it reaches 280°.

The results achieved with this algorithm can be compared to those obtained using a baseline SAC. When the robot starts without contact, this algorithm requires a long time to explore the space without finding actions to manipulate the rod. To ensure that the robot approaches the rod, a distance term can be added to the reward. In the proposed method, the reward is easier to design because part of the complexity is delegated to the expert trajectories and the definition of the contact configurations that the robot must visit to solve the task. Once the contact with the rod is made, the SAC algorithm is unable to reconfigure the contacts in less than 200000 iterations.

4.2.4 Discussion

Use of configuration spaces in the learning

The previous results demonstrate that it is possible to integrate prior knowledge into the learning process by defining a knowledge graph based on contact configuration spaces. This graph allows for greater sample efficiency while maintaining the flexibility brought by learning.

This method provides interpretability during the learning process. Examining the results of the Selector and the choices of the Evaluator allows to determine the agent's current configuration space and the targeted configuration space to solve the task. The CartStemContact first navigates to the space containing the goal and then solves the task within that configuration space. The Evaluator of the RodManipulator employs external agents to reconfigure the contacts.

Reusability

The Evaluator utilizes both external and internal agents to solve the task at hand. It employs internal agents to correct the results of external agents. The example of the CartStemContact demonstrates that using learned or not learned internal agents does not lead to a decrease in performance, as the entire network has the same behavior. The two examples show the possibility of reusing and combining pre-learned, learned, or not learned agents. This validates the use of a task-independent structure as the basis of the learning process. The Evaluator can leverage the knowledge of external agents to solve a given task more efficiently.

Combine classical learning and prior knowledge

Optimisation-based algorithms can only solve a task if the goal belongs to the same convex space as the effector's one. The CartStemContact shows that we can continue to use the advantage of optimisation-based algorithms by learning the transition between two configuration spaces, even if they correspond to different convex spaces. This has two benefits: it can improve general sample efficiency and overcome the limitations of a convex approach. However, this study should be extended to include simultaneously learned and not learned internal agents. This would be particularly interesting in the case of the RodManipulator, where two opposite contact configurations can be handled using convex optimisation, whereas the other configurations are learned.

Another prior knowledge is provided by the expert trajectories. Unlike Behavior Cloning methods or other approaches that use entire trajectories, the definition of the knowledge graph and sub-part of the trajectories guide the learning by defining the passage point necessary to achieve the desired behavior. In the RodManipulator example, the use of expert trajectories to reconfigure contacts is easier to obtain than Behavior Cloning because it is independent of the task to be solved and does not correspond to an entire episode. Although this trajectory is not optimal, the algorithm can use it to solve the task and improve the efficiency of its own strategy.

Overall method

We propose to use the contact configurations to create the configuration spaces, motivated by the importance of optimisation approaches in soft robotics. However, other approaches could also be considered, such as automatically defining the configuration spaces based on the robot's workspace in a specific contact configuration. This method is an intuitive way to incorporate prior knowledge into learning algorithms. Even though it can be applied to other fields than soft robotics, it is particularly relevant for the control of soft robots as traditional methods for path planning and predictive control with contact are limited.

The complexity of the general structure of the network has some limits. Each agent corresponds to one network, increasing the number of hyperparameters and update at each time step. Moreover, the definition of the configuration spaces can influence the convergence speed. In fact, agents share information, and an under-exploited agent takes more time to learn and thus to provide reliable information. There are several ways to improve this method. First, it would be interesting to extend the application domain of this technique, for example, by using multi-task and dynamic graph. With a method for dynamically learning configuration spaces, these spaces will be useful for the task and will be built to solve the task. In addition, the more multi-task objectives the robot has to solve, the more general the configuration spaces will be. Modifications of the learning conditions could also be considered, such as local updates of the networks when the number of configuration spaces becomes large or the reward of the external agents depending on the length of the sequence of action.

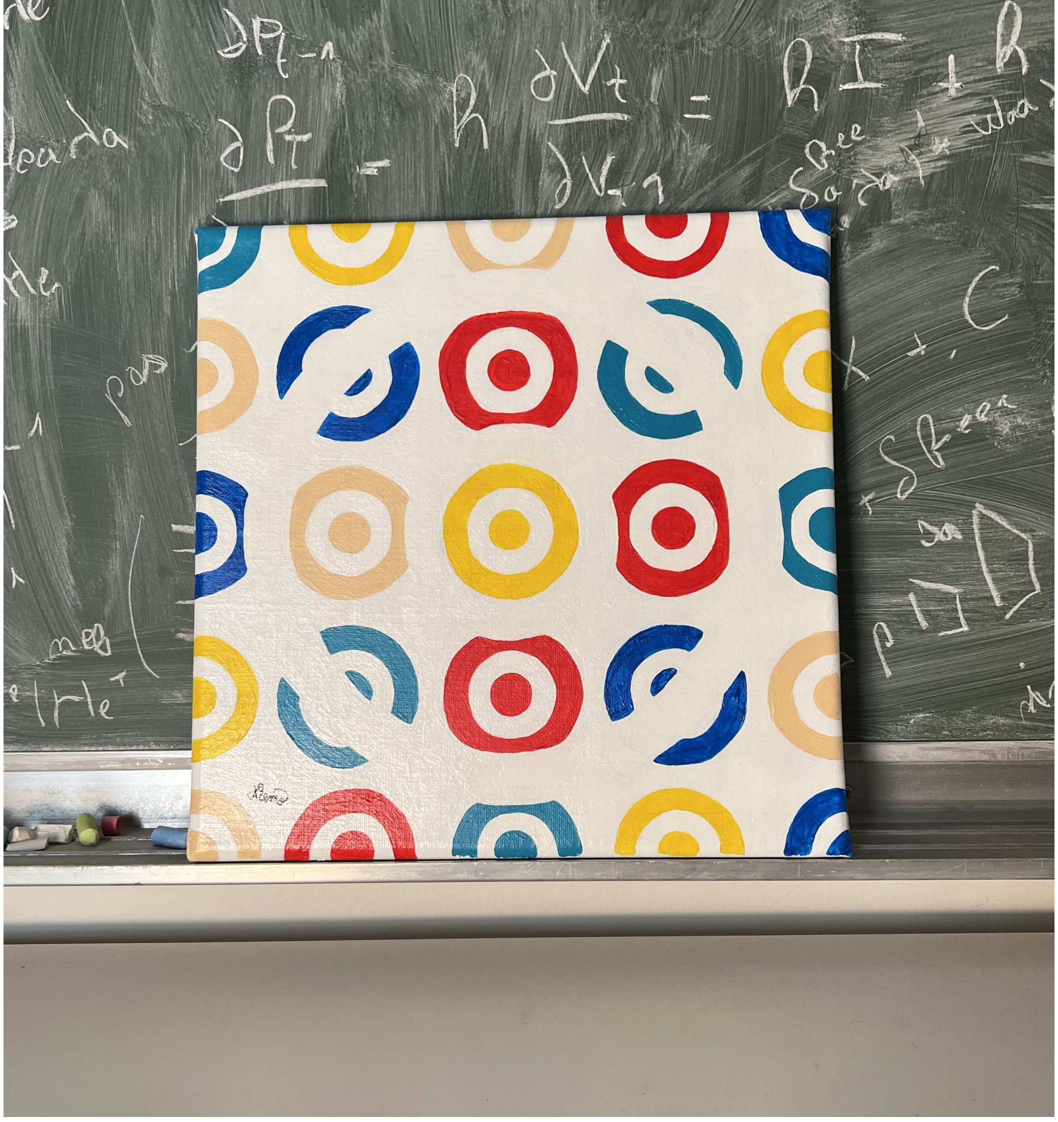
4.3 Conclusion

In this Chapter, we investigate two methods for solving high-level tasks using learning while maintaining the efficiency of inverse models and optimisation. Both methods describe the state of a robot in a specific manner. The proxy approach involves describing the robot in a reduced configuration space by using a fast model extracted from the FEM model. This proxy can be used to generate a high-level trajectory for solving a task using RL algorithms, which can then be corrected and transferred to the FEM model using an inverse model and optimisation. This motion planning method can be applied to complex soft robotics systems, and has successfully solved manipulation and locomotion tasks. The graph approach classifies the robot state in different configuration spaces based on the observation of contact configurations. These spaces provide prior knowledge for learning and can be used to combine learning and optimisation approaches. If the configuration of the robot is in a convex space, convex optimisation can be used to control it efficiently. RL algorithms can be used if the robot has to move to another space, for example, to reconfigure the contacts.

However, these methods have limitations. There is a loss of expressiveness in learning. The configuration spaces graph method requires to construct the graph, which can be difficult when there are many contacts and these contacts are difficult to localize. The proxy model is a simplified version of the FEM, and learning algorithms can find biased solutions by exploiting the differences between the models. The discussion sections suggest some ideas for dealing with these limitations. The methods proposed in this chapter are therefore promising for control in soft robotics. Another limitation is that all the tests were carried out in simulation. In order to fully validate these methods, they must also be transferred to physical robots. This transfer has its own difficulties, such as the quality of the FEM model for the proxy method. In the graph method, detection of contact configurations from the sensors is necessary to use it on a physical prototype.

5 Condensed FEM model for control and design in soft robotic applications

Contents



6. Communion.

— **Nemo**

We live together, and we all resemble one another. We develop a collective identity based on shared ideas. Although our individuality is distinct, it is intertwined with the larger whole. This individuality is expressed through words and ideas that resonate with others. We often return to the same ideas and words. This is a matter of doing something different from them.

In previous chapters, various methods for describing the state of a soft robot are explored. On one hand, FEM is used to describe the state of the robot (Chapter 3), while on the other hand, simplified descriptions of the state are proposed using a proxy model or by dividing the observation space into several configuration spaces (Chapter 4). All these methods can be used to solve high-level tasks, with respect to the underlying modelling assumptions. The objective of this Chapter is to propose an intermediate method based on a reduced description that retains the expressiveness of the FEM model. We propose to use the condensed FEM model presented for one-step control (see Section 2.2.3) to implement this method. The condensed FEM model represents the kinematics or dynamics of the robot in the space of constraints (actuators, effectors, and contacts). It can be used in several ways.

First, the condensed model can be learned. FEM modelling is a powerful tool for modelling the behavior of soft robots; however, its use in real time can be difficult for non-specialists in numerical computation. Machine learning approaches enable rapid prediction of robot behavior but use the robot's mechanics as a "black box", particularly for generating training data. The approach developed in the first section is a "white box" approach, in which the mechanical structure of the model is used to perform targeted learning (5.1.2). This approach preserves the mechanical properties of the predicted quantities for contact (5.1.4) and non-contact (5.1.3) control and for design applications (5.1.6). It provides real-time prediction capability for embedded control (5.1.5). In the second section, the condensed model is used for trajectory optimisation with soft robots (5.2). Trajectory optimisation and multi-time-step control of soft robots using FEM models and optimisation methods is challenging because of the high dimension of the manipulated mathematical systems. For instance, using the optimal control formalism presented in Section 2.2.3 with the robot's FEM model involves solving QP sequences with matrices of sizes proportional to the size of the simulation mesh. The condensed model resolves this dimensional problem by using small matrices in the backward phase of DDP.

5.1 Learned condensed FEM model

In this section, we present a framework to learn a compact mechanical representation of a soft robot based on the condensed FEM model. This model can be used in a quasi-static inverse control of soft robots and handle contact management. We highlight its modularity by showing that the control of a complete gripper on a manipulation task can be performed using the condensed FEM model learned on a single soft finger. Then, we evaluate the low memory consumption of this model. We show it is suitable for embedded control on the case of the inverse control of a physical pneumatically actuated soft Trunk robot. Finally, we extend the condensed FEM modelling to manage design parameters and show how we can use its inherent differentiability in both calibration and design optimization applications. The proposed framework and contributions are summarized in Figure 5.1.

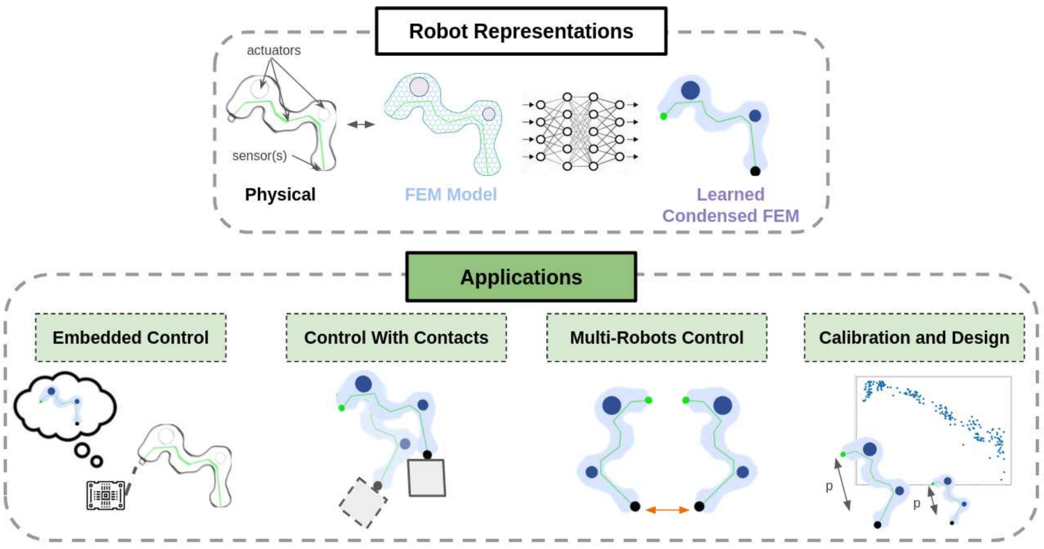


Fig. 5.1: Illustration of the proposed framework and its applications. A physical soft robot is modeled using the FEM. The condensed modeling is obtained by projecting the FEM characteristic matrices in the constraint space. A neural network is trained offline for predicting the condensed FEM modeling from both the initial robot condensed state and applied actuation. For a given robot, a single learned condensed FEM modeling may be used in several applications i.e. for real-time embedded control, for inverse control involving a set of predefined contact points, for controlling several coupled similar robots altogether as well as for calibration and design optimization.

5.1.1 Condensed FEM Modelling

As we explained in Section 2.1.3, the mechanical behavior of soft robot deformations can be described through continuum mechanics. In this section the quasi-static formulation is consider:

$$K(q)\Delta q = f_{ext} - f_{int}(q) + H_a^T \lambda_a + H_c^T \lambda_c \tag{5.1}$$

where Δq are small displacement of nodes around their current position q. We consider elastic behaviours expressed from a Poisson ratio ν and a Young Modulus E though in principle the condensed FEM model could be learned for any elastic model. In addition, the definition of δ_c changes slightly from that given in Section 2.1.3, because we are now considering probable contact zones: at the learning stage, we do not yet have an obstacle, or a force applied by another object on the contact zone. δ_c can not be defined as the signed distance between two objects. Instead, a displacement δ_c of the contact points relative to their rest position is imposed, in order to reproduce the effect of an obstacle on the contact zone.

A robot can be modelled using its inputs and outputs. The one-timstep control proposed in the Section 2.2.3 introduces the projection of the dynamics into the constraint space. The resulting mechanical quantities gather the mechanical coupling between passive constraints (effectors e) and active constraints (actuators a and contacts c). This is a method of representing the inputs and outputs of the system while keeping the expressiveness of the underlying FEM model. It is possible to write these projected equations in the quasi-static context. The integrated impedance matrix A is then replaced by the stifness matrix K. In this case, $W_{ij} = H_i K^{-1}(x) H_j^T$ for $i, j \in \{e, a, c\}$ is the compliance (inverse of the stiffness) projected into the effector, actuator and/or contact spaces. Both the $W_{ij} = H_i K^{-1}(x) H_j^T$ for $i, j \in e, a, c$ and δ_i^{free} for $i \in \{a, c, e\}$ could be seen as a reduced model of the robot as illustrated in Figure 5.2.

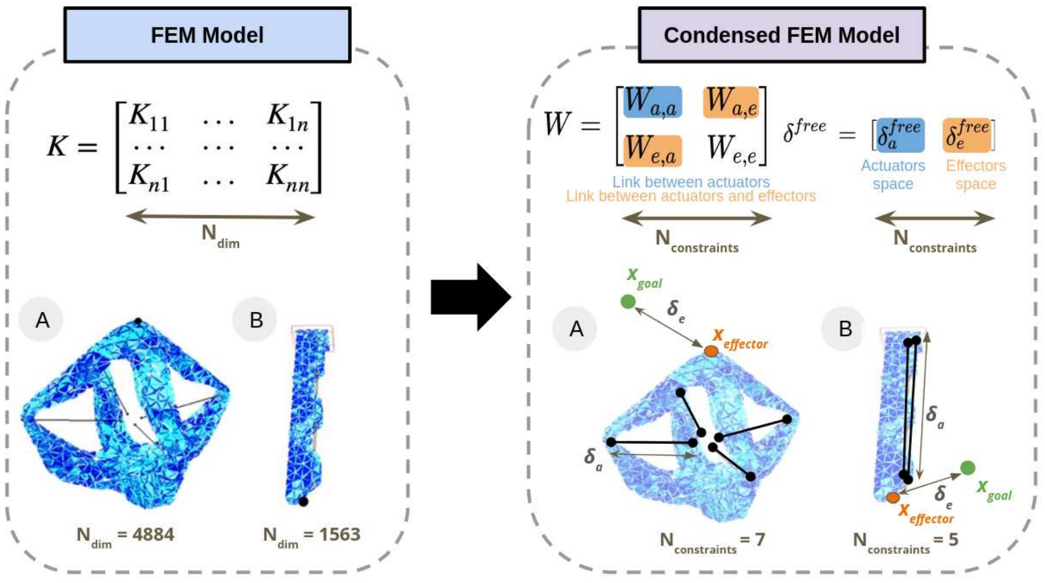

FEM Model
$K = \begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \cdots & \cdots & \cdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix}$
N_{dim}
A
B
$N_{dim} = 4884$
$N_{dim} = 1563$
Condensed FEM Model
$W = \begin{bmatrix} W_{a,a} & W_{a,e} \\ W_{e,a} & W_{e,e} \end{bmatrix}$
$\delta^{free} = [\delta_a^{free} \; \delta_e^{free}]$
Link between actuators
Link between actuators and effectors
Actuators space
Effectors space
$N_{constraints}$
$N_{constraints}$
x_{goal}
δ_e
$x_{effector}$
δ_a
A
B
$N_{constraints} = 7$
$N_{constraints} = 5$

Fig. 5.2: Illustration of the condensed FEM model for both the Diamond and Soft Finger robots. They are respectively actuated by 4 and 2 cables. In both cases, a FEM model of the robot (left) is used to learn a condensed model (right). The reduced model is based on the projection of the compliance matrix into the constraint space, resulting in drastically reducing the number of characteristic dimensions describing a robot state.

In an inverse problem, this reduced model can be used to control soft robots (see Section 2.2.3). In the direct problem, when the actuator forces intensity $\Delta\lambda_a$ is varying, then the corresponding motion of the effectors $\Delta\delta_e = W_{ea}\Delta\lambda_a$ can be computed. As actuators are also controlled in displacement, varying the position of the actuators $\Delta\delta_a$ lead to obtain the motion created on the effectors through $\Delta\delta_e = W_{ea}W_{aa}^{-1}\Delta\delta_a$. This equation provides the Jacobian $J = W_{ea}W_{aa}^{-1}$ of the soft robot direct kinematics. It shows that both direct and inverse models of soft robots can be derived as long as an approximation of equations 2.14 giving the effective state of the actuation δ_a and the contact δ_c are available.

5.1.2 Learned Condensed FEM modelling

In this section, the learned condensed FEM model is introduced. The different applications tackled with the learned condensed FEM model are also presented, namely modelling and control with and without contacts, embedded control and design optimization.

Mechanical Representation Learning

We suppose that a high dimensional state of the robot is available through simulation. For instance, when using tetrahedral FEM with linear interpolation, the state of the robot is the position of all the vertices of its mesh. For reducing the dimension of the robot state, the global compliance matrix projected in the constraint space W (see section 5.1.1) is a good candidate because it is a small dimensional matrix, independent of the size of the FEM mesh and it expresses a direct relationship between constraints. In combination with the free displacement vector δ^{free} (see Section 2.2.3), it is relevant enough for modelling and controlling a soft robot. In theory, W depends on the global state of the FEM model. However, in the case of a deformable robot, the deformations are due to the state of the actuators and contacts on the robot. It is therefore possible to obtain W only for states that can be reached by the robot, which depend on the state of the contacts and actuators. The computation of W is expensive because it involves the multiplication and inversion of several large matrices and this needs to be computed at every simulation step. This is an incentive for learning it rather than computing it from mechanical simulation. Learning these mechanical quantities rather than non-supervised ones makes it possible to obtain interpretative robot representations useful for simulating, controlling and even design the robot using mechanical equations.

The learning problem is then resumed as the following. Let δ_a, δ_c, W and δ^{free} be respectively the effective states reached by both actuation and contact constraints when applying a specific active constraint state as well as corresponding projected compliance matrix and free displacement of the robot. The goal is to reconstruct (W, δ^{free}) from initial values (W_0, δ_0^{free}) when $\lambda_a = \mathbf{0}$ and $\lambda_c = \mathbf{0}$, and current active constraint states δ_a and δ_c. This leads to learn a function F such that

$$\widetilde{W}, \widetilde{\delta}^{free} = F(\delta_a, \delta_c, W_0, \delta_0^{free}) \tag{5.2}$$

W_0 is a good compliance initialization since it already captures the mechanical coupling between constraints and the compliance matrix varies smoothly from that initial position in the workspace of the robot. This mathematical formulation excludes cases when the robot may have several configurations for a same actuation. This could happen when using highly non linear material models or when considering non-linear phenomenons such as buckling.

Learning Model

The compliance matrix W is a symmetric matrix that represents the relations between the different constraints. It is formed by the mechanical matrices introduced in Eq. 2.14:

$$W = \begin{bmatrix} W_{ee} & W_{ea} & W_{ec} \\ W_{ae} & W_{aa} & W_{ac} \\ W_{ce} & W_{ca} & W_{cc} \end{bmatrix} \tag{5.3}$$

The magnitude of both the values of W and δ^{free} depends of both the type of constraints and the measurement units chosen for the simulation. For avoiding vanishing gradients during the learning process, both the input and output data are element-wise standardized. This standardization is computed once using all the training set, using the formula:

$$\mathbf{X} = \frac{X - X_{mean}}{X_{std} + \epsilon} \tag{5.4}$$

where X is the tensor $(\delta_a, \delta_c, W, \delta^{free})$, X_{mean} and X_{std} are respectively the mean and standard deviation of the training set, ϵ is a small quantities to avoid computational singularity and $\mathbf{X}$ is the standardized data. In order not to overload the notations, standardized inputs for learning and not standardized inputs for control are systematically considered afterwards.

As the matrix is symmetric, the networks are trained in a supervised manner through minimizing the reconstruction loss:

$$L = \sum_{\delta_a, \delta_c} [||\widetilde{W}^{tri} - W^{tri}||^2 + ||\widetilde{\delta}^{free} - \delta^{free}||^2] \tag{5.5}$$

where W^{tri} is the upper triangular part of the matrix W, $\widetilde{W}^{tri}$ and $\widetilde{\delta}^{free}$ are the predicted values of W^{tri} and δ^{free} as given by the network for actuation displacement δ_a and contact displacement δ_c. As all data are standardized, the computed loss function obtained on different training data are somewhat comparable. MLP are sufficient for learning a mechanical representation for control and design tasks. Input and output of the network are vectorized data.

As the framework is a supervised learning framework, pre-computation steps where data are collected and the learning model is trained are first performed. The data $(\delta_a, \delta_c, W^{tri}, \delta^{free})$ are obtained through simulation and stored in a database. Each data point is associated to a different robot configuration after applying an active constraint displacement (δ_a, δ_c). The range of possible values of both δ_a and δ_c are bounded in continuous intervals. Sampling the active constraint space is done using Scrambled Halton sampling algorithm [Mas04] for building the training dataset and random search for the test dataset. Compared to homogeneous grid search strategies, it enables a homogeneous and better filling of a high dimensional sampling space and therefore a better scalability to robot having numerous actuators and contacts. The number of points of the test set is chosen to amount for 25% of the number of points of the training set. Once the data is acquired, learning consists in training the network to predict the quantities (W^{tri}, δ^{free}) from the quantities $(\delta_a, \delta_c, W_0^{tri}, \delta_0^{free})$ using MLP and Eq. 5.5. Unless mentioned otherwise, the used network in the experiments introduced in this section is a fully connected MLP with 3 hidden layers of 450 nodes each. Regarding the learning rate, it is initialized to 10^{-4}. In all our experiments, an adaptative scheduler is systematically applied for reducing the learning rate when the optimizer falls in a local optima.

Control applications

Once learned, the condensed FEM model is used in a control loop as described in the following workflow:

1. δ_a, δ_c, W_0^{tri} and δ_0^{free} are recovered from FEM simulation. During the first iteration of a soft robot control experiment, initial δ_a and δ_c are obtained for zero active constraints efforts applied on both the actuators and contacts.

2. Prediction of $(\widetilde{W}^{tri}, \widetilde{\delta}^{free}) = F(\delta_a, \delta_c, W_0^{tri}, \delta_0^{free})$ using the trained network. $\widetilde{W}$ is reconstructed from $\widetilde{W}^{tri}$. Although the condensed FEM model is learned

on a predefined set of fixed goals p_{goal}^{train}, one for each of the effectors, it can be easily adapted to other goals p_{goal} during the control phase with the formula:

$$\delta_e^{free} = \widetilde{\delta}_e^{free} + p_{goal}^{train} - p_{goal} \tag{5.6}$$

3. An inverse optimization problem such as the one from Eq. 2.15 is solved using a quadratic optimization solver. The aim is to compute the actuation effort vector λ_a for controlling the robot. The C++ library Prox-QP [Bam+22] is used for implementing the solver in our work.

4. The actuation effort λ_a is applied on the robot and the respective new actuation and contact states δ_a and δ_c are recovered. At this step, corrective motion can be applied for improving performance, stability and precision. In particular, it enables to palliate the two main sources of errors which are due both from the learning and the simulation-to-reality gap.

5. The control loop starts again from step 1 until convergence, that is until an accuracy or stability criterion is satisfied.

In sections 5.1.3 and 5.1.4, this pipeline is applied respectively without and with contacts on simulated robots. As the simulation-to-reality transfer of the robots considered in our experiments has already been studied in previous works, the simulation could be considered as a ground-truth for assessing our algorithms. Constraints states could then be directly retrieved as results from the simulation. In section 5.1.5, this workflow is applied on a physical prototype. Constraint states are retrieved from a set of sensors.

Calibration and Design Optimization applications

In section 5.1.6, it is shown that both W_0 and δ_0^{free} are good characterizations of the design of a soft robot. It is therefore possible to build a differentiable link between design parameters P and these quantities as follows:

$$\widetilde{W}_0, \widetilde{\delta}_0^{free} = G(P) \tag{5.7}$$

where G is a MLP network. Data are collected in the same way as for the network F and the training loss is similar to 5.5.

Then applications such as calibration and design optimization are explored. When design parameters are relevant, the condensed FEM model is extended following:

$$\widetilde{W}, \widetilde{\delta}^{free} = F(\delta_a, \delta_c, G(P)) \tag{5.8}$$

This enables to build a differentiable link between design parameters and mechanical quantities in given active constraint states. Fitness functions are then built from these mechanical quantities. The entire learning pipeline with design parameters is illustrated in Figure 5.3.

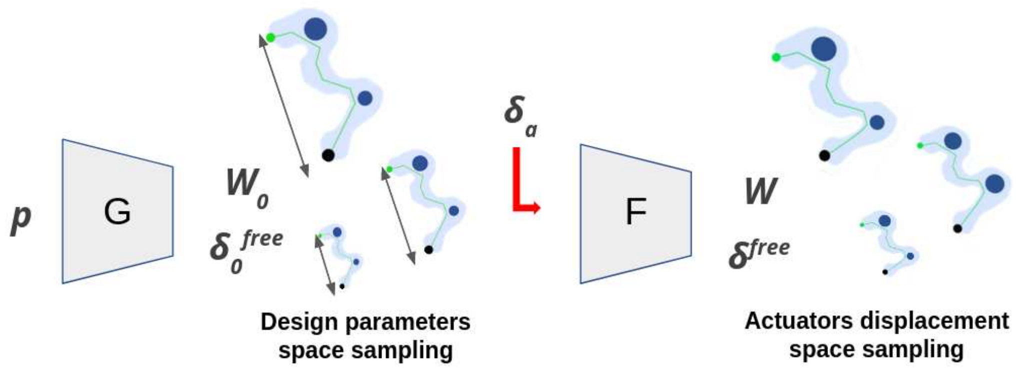

p
G
W0
δ0 free
δa
F
W
δfree
Design parameters space sampling
Actuators displacement space sampling

Fig. 5.3: Illustration of the pipeline for learning mechanical states of Soft Robot. G predicts the mechanical state of the robot in its rest position from design parameters, and F predicts the mechanical state of the robot from both G outputs and a given actuation displacement.

5.1.3 Soft robot control using a condensed FEM modelling without contacts

Following the introduction of the condensed FEM model, a set of experiments are introduced below for highlighting control applications without taking contacts into account. The open-loop control is illustrated and validated using two FEM models of different soft robots driven by cables, the Diamond robot (see section 3.1.2) and a Soft Finger robot (see section 3.1.2). The simulation mesh of the Diamond robot is made of 4147 tetrahedrons. The Soft Finger robot is made up of three segments connected by accordion-shaped joints and actuated by 2 cables. Its simulation mesh is made of 1557 tetrahedrons.

For each robot, the data are retrieved from the simulation as described in subsection 5.1.2. Statistics about the data, the final loss and the best epoch[1] are

[1]An epoch corresponds to the training of the network with all the training data for one cycle.

given in Table 5.1. During data acquisition, the space of displacements, that is the space of relative lengths δ_a imposed on the cables, is discretized and the actuation state is observed as the reached length of the cables at equilibrium.

Tab. 5.1: Data acquisition parameters for each robot for $n_l = 6500$ sample points. Cables displacement are sampled in $[\delta_{a,min}, \delta_{a,max}]$ (mm). Final test loss values for Diamond and Soft Finger robots for 9000 learning iterations. The best trained network encountered is kept at best epoch .

Robot	Cables		Final Loss	Best Epoch
	$\delta_{a,min}$	$\delta_{a,max}$		
Diamond	0	30	6.52e-5	8960
Finger	0	20	3.79e-6	8850

Open Loop control of a single robot from a learned condensed FEM modelling

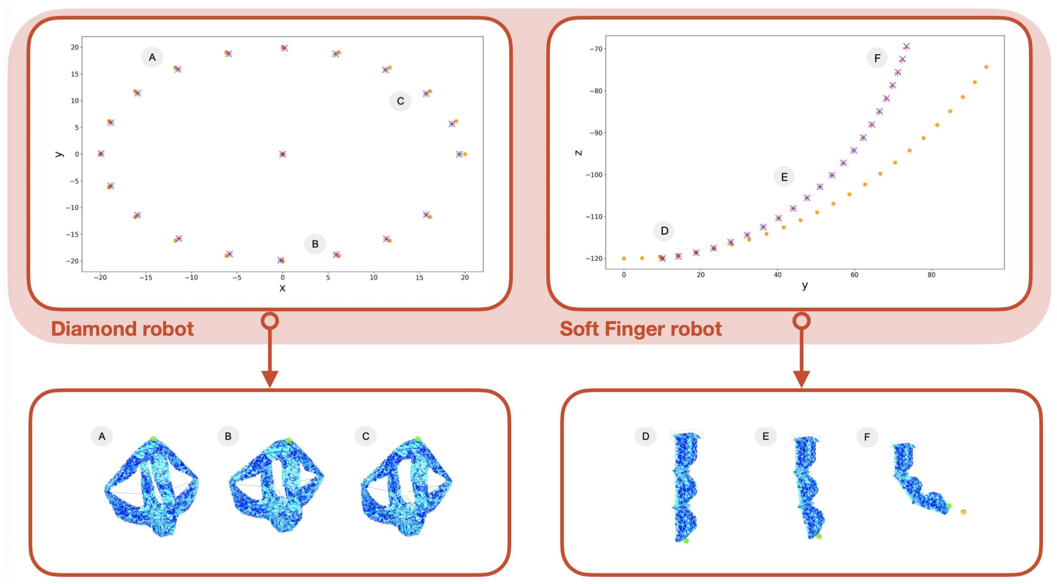


Fig. 5.4: Control results for the Diamond and Soft Finger robots without contact. A-B-C) Trajectory of the end effector of the Diamond robot in the (x, y) plane (mm) for 30 different goals (orange dots) when using learned mechanical matrices (blue cross) or computed mechanical matrices (red cross). 3 different position of the Diamond robot along this trajectory are shown as examples. The effector is represented by a green dot, and the goal by an orange dot. D-E-F) Trajectory of the end effector of the Soft Finger robot in the (y, z) plane (mm) for 25 different goals (orange dots) when using learned mechanical matrices (blue cross) or computed mechanical matrices (red cross). 3 different position of the Finger robot along this trajectory are shown as examples. The effector is represented by a green dot, and the goal by an orange dot.

For evaluating the learning process, the average loss is computed on the test set of random robots configurations. Learning results for the condensed FEM model are given in Table 5.1. Although there is more constraints in the Diamond problem, it takes the same amount of data samples for training the MLP for the Diamond and the Soft Finger. This is because the projected mechanical matrices relative to the Soft Finger robot varies much more than that relative to the Diamond robot. In fact, most non-linearities in mechanics come from rotations, and in the Finger example there are more rotations than in the Diamond example. For the Soft Finger, the influence of the actuators on the end effector changes a lot depending on the configuration of the robot, that is depending on how bent the robot is. On the contrary, the effector always has similar behavior in terms of constraints for the Diamond.

In order to evaluate the use of the learned condensed FEM model for inverse modelling, a trajectory is considered in the form of several goal positions to be reached by the effector for each robot. As the soft robot dynamics is neglected in this study, the results are only valid for trajectories time-scales in the quasi-static regime. The results are shown in Figure 5.4.

A circular trajectory of the effector in a horizontal plane is considered for the Diamond robot. This trajectory is chosen for being convenient for two dimensions visualization although the robot end effector can be controlled in all three directions of space. Then the positions obtained for each goal using the condensed FEM model learning based control pipeline and the classical inverse model are compared. In most configurations, the two approches lead to similar positions of the end effector. This shows that the condensed model is a good representation of the system. On some positions, small-scale errors between the learned model and the full model can occur: this comes from the learning errors specific to the method. This error can be further decreased by increasing the discretization of the actuation space during data collection and by judicious choices of networks hyper-parameters. For the Soft Finger robot, a trajectory is chosen to force the robot to bend. The trajectory is not in the robot's workspace. In such case, the optimization will minimize the distance between the end effector (in the workspace) and the goal (in the trajectory). However, both control pipeline using the learned condensed FEM model and classical FEM model achieve similar results. The same conclusions as for the Diamond robot are valid for the origin of these errors and how they can be minimized.

Investigation on mesh influence

Increasing the fineness of the mesh generally makes it possible to obtain more precise simulations at the cost of significant computation time. A compromise between speed and precision has generally to be found for enabling to control a soft robot in real time. This can be critical for robots with complex behavior for which both precision and speed are determining factors. As the condensed FEM model is built from the projection of mechanical matrices in constraint space, its size is independent of the mesh size, which makes it a good candidate to satisfy both of these criteria at the cost of more computationally expensive offline simulations. The following experiment highlights the property of the condensed FEM model which speed of use in a control loop is independent in time of the size of the mesh.

Three condensed FEM models of the Soft Finger are trained for different mesh sizes. The three Soft Fingers mesh discretization as well as simulation performances are introduced in Figure 5.5.

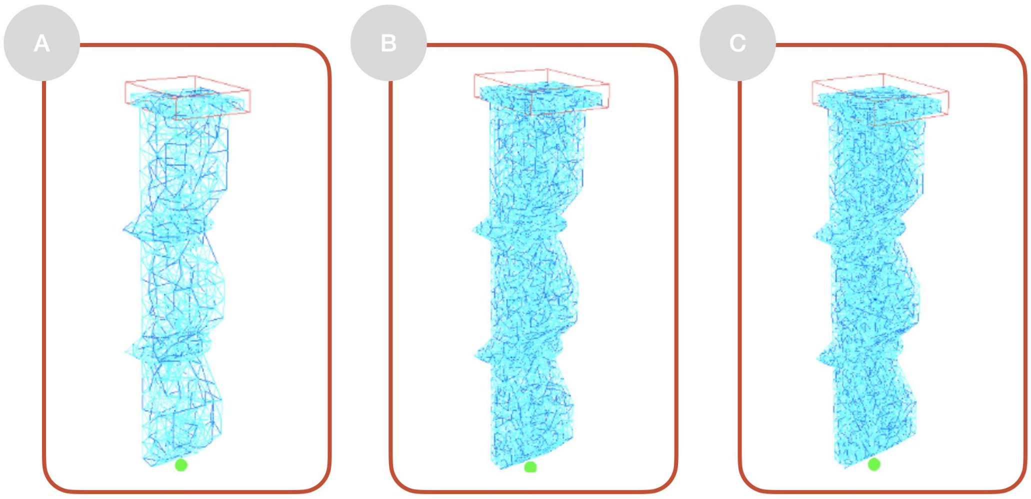


Fig. 5.5: Soft Fingers with different mesh resolutions. For each of them, the mesh resolution in terms of number of tetrahedrons and number of frames per second (FPS) of the simulation is indicated. A) Basic Soft Finger: 1557 tetrahedrons, 32.6 FPS. B) Fine Soft Finger: 8895 tetrahedrons, 4.8 FPS. C) Super Fine Soft Finger: 17852 tetrahedrons, 1.8 FPS.

For being able to compare them, 6500 Soft Finger states are sampled for building the training set and a fixed number of 9000 training iterations is performed for each Soft Finger. Learning results are summarized in Table 5.2. Magnitude order of final training losses are similar.

Tab. 5.2: Final test loss values and FPS of the simulation when using the condensed FEM model for the 3 Soft Finger robots with different meshes discretizations. The best trained network encountered is kept at each epoch.

	Final Loss	Best Epoch	FPS with Learned Model
Soft Finger	3.79×10^{-6}	8850	308.6
Fine Soft Finger	4.26×10^{-6}	8900	334.4
Super Fine Soft Finger	3.60×10^{-6}	8990	296.7

Whatever the size of the mesh, calling the learned neural network for getting a given mechanical state takes the same amount of time. All three models perform equally well on the considered trajectory. The reduced model does not depend on the mesh size, but on the number of constraints in the problem. In this experiment, only the simulation is considered. In most cases, the chosen hardware also limits the running frequency. When the fineness of the mesh starts to become the blocking factor, the condensed FEM model enables constant time predictions compatible with real-time control.

5.1.4 Soft robot control using a condensed FEM modelling with contacts

The integration of contacts into the method is illustrated using the Soft Finger robot. It is assumed that contacts are located at three fixed points on the surface of the Soft Finger. For the moment, whatever the example, the method only works if the contact points can be identified in advance. The robot is in contact with a perfect rigid object. To avoid having to solve Coulomb's and Signorini's laws, the simplified assumption that the Soft Finger contacts points share the same location than the associated contact points on the object is taken. This assumption facilitates the optimization problem, but it does not change the representation of contacts in the condensed FEM model.

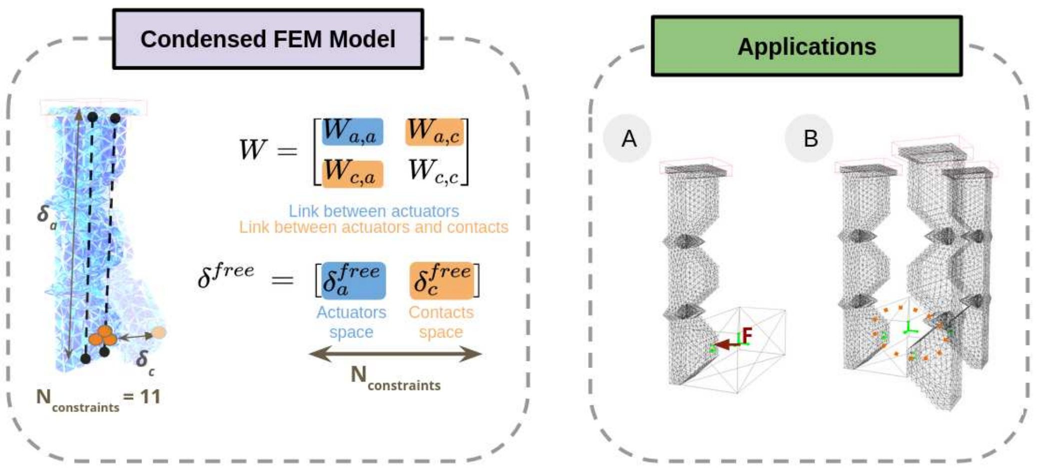

Condensed FEM Model
δ_a
δ_c
$N_{constraints} = 11$
$W = \begin{bmatrix} W_{a,a} & W_{a,c} \\ W_{c,a} & W_{c,c} \end{bmatrix}$
Link between actuators
Link between actuators and contacts
$\delta^{free} = [\delta_a^{free} \ \delta_c^{free}]$
Actuators space
Contacts space
$N_{constraints}$
Applications
A
B
F

Fig. 5.6: Illustration of the condensed FEM model for a Soft Finger with contacts and the two considered control scenarios. The Soft Finger is actuated by 2 cables and has 3 fixed contact points located at the tip. The scenarios are A) a Soft Finger robot pushing a button, and B) a Soft Gripper robot manipulating a cube. The cube effector is represented by a green 6D frame.

Two application cases are considered and illustrated in Figure 5.6:

- A Soft Finger pushing a button shaped like a cube. The resistance of the button is modeled as a constant horizontal force. This example is used for validating the learning of the various mechanical quantities of the condensed FEM model related to contacts i.e. W_{cc}, W_{ca} and δ_c^{free}.

- A Soft Gripper handling a cube. The gripper consists of 3 identical Soft Finger robots. This example shows that a single condensed FEM model learned for a Soft Finger robot can be used in another inverse optimization scheme coupling several Soft Fingers.

The introduction of contact constraints increases the dimension of the constraints space. In the Finger example, there are two actuators (2 constraints) and three contact points (9 constraints). Exploring this space is exponentially complex, in terms of constraints. Moreover, using a box sampling on both actuation and contact constraint spaces, as for both the Diamond and Soft Finger without contact from section 5.1.3, would result in many robot state that are irrelevant in our applications. That is why in both cases, the dataset is generated using an inverse control scheme and sampling in the cube effector displacement space instead. Statistics about the datasets and the learning are given in Table 5.3.

Tab. 5.3: Data acquisition parameters and final loss for Soft Finger and Soft Gripper robots in the contact case. The number of training epochs for the Soft Finger and the Soft Gripper are respectively 50000 and 42540.

Robot	Cube (mm)			Final loss	Best Epoch
	$\delta_e^{cube,x}$	$\delta_e^{cube,y}$	$\delta_e^{cube,z}$		
Soft Finger	[-2, 2]	[33, 37]	[-95, -94]	1.67e-6	42540
Soft Gripper	[-2, 2]	[33, 37]	[-95, -94]	1.24e-3	38440

Inverse problem for interaction between object and soft finger

A simplified pipeline for taking into account contacts in the optimization process, without needing to solve the Signorini's and Coulomb's law is now developed.

Let $X_{cube} \in \mathbb{R}^6$ be the location of the cubic object in both translation and rotation. Let ^{prev} be the notation used to refer to quantities calculated at the previous time step in an iterative process. We want to control for both the translation and orientation of the cube with the help of a set of Fingers. It is then equivalent to minimize the following quantity:

$$\min_{\lambda_a} ||X_{cube} - X_{cube}^{goal}||^2 \tag{5.9}$$

where X_{cube}^{goal} is the goal location to reach and $||.||$ is the euclidean norm. To solve this problem, X_{cube} has to be written as a function depending on λ_a. This expression is obtained from the kinematics relations written for the cubic object and the condensed mechanics applied to the Soft Fingers. Limitations on the actuation forces are added afterward, when solving the complete inverse problem.

Kinematics relations between points fixed on a rigid object: From the hypothesis that contact points location on both the Soft Finger and the object are shared, it results that $\lambda_c = \lambda_{c,finger} = -\lambda_{c,cube}$ where $x_{c,finger}$ and $x_{c,cube}$ are the euclidean positions of the matching contact points on the interface between the Soft Finger and the cube.

Using the Varignon formula on the cube, the location X_{cube} and $x_{c,cube}$ are linked as:

$$x_{c,cube} - x_{c,cube}^{prev} = J_c(X_{cube} - X_{cube}^{prev}) \tag{5.10}$$

where $J_c \in \mathbb{R}^{N\times 6}$ is the Jacobian matrix linking the kinematics of the manipulated object to the contact points and N the number of contact point. For one point, the Jacobian matrix is given by:

$$J_c = \begin{bmatrix} 1 & 0 & 0 & 0 & -C_z & C_y \\ 0 & 1 & 0 & C_z & 0 & -C_x \\ 0 & 0 & 1 & -C_y & C_x & 0 \end{bmatrix} \tag{5.11}$$

where $C_i = X_{cube,i} - x_{c,cube,i} : i \in [x, y, z]$ are the coordinates of the vector $\vec{C}$ as illustrated in Figure 5.7. Generalization for several contact points is straightforward.

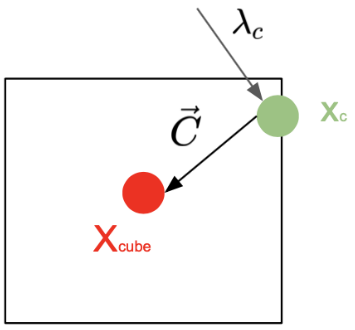


Fig. 5.7: Illustration of the quantities for computing the Jacobian J_c for one contact point.

From this Jacobian matrix the resulting contact force on the cube is then expressed as $T_{cube} = J_c^T \lambda_c$.

Condensed mechanics applied on a Soft Manipulator: By isolating the robot and writing the condensed mechanics according to Eq. 2.14, the values of x_c are expressed from λ_a, λ_c and the free displacement δ_c^{free} of the contact point as:

$$x_{c,cube} = W_{ca}\lambda_a + W_{cc}\lambda_c +_c^{free} \tag{5.12}$$

By integrating these results into Eq. 5.10 and multiplying by $D = J_c^T W_{cc}^{-1}$, we obtain:

$$\begin{aligned} &Dx_{c,cube}^{prev} + DJ_c(X_{cube} - X_{cube}^{prev}) = D\delta_c^{free} + DW_{cc}\lambda_c + DW_{ca}\lambda_a \\ \iff &DJ_cX_{cube} = DW_{ca}\lambda_a + [D\delta_c^{free} + DW_{cc}\lambda_c + DJ_cX_{cube}^{prev} - Dx_{c,cube}^{prev}] \\ \iff &X_{cube} = \underbrace{[DJ_c]^{-1}DW_{ca}}_{\mathbb{A}}\lambda_a + \underbrace{[DJ_c]^{-1}[D\delta_c^{free} + DW_{cc}\lambda_c + DJ_cX_{cube}^{prev} - Dx_{c,cube}^{prev}]}_{\mathbb{B}} \end{aligned} \tag{5.13}$$

which is of the form $X_{cube} = \mathbb{A}\lambda_a + \mathbb{B}$ if we suppose that $DW_{cc}\lambda_c = T_{cube}$ the contact forces are known. The optimization objective from Eq. 5.9 is now well posed.

Both the Soft Finger robot and the Soft Gripper robot differ in the way the force T_{cube} is computed. From the new values of X_{cube} and λ_a, it is then possible to compute a new value of λ_c using Eq. 5.13.

Application: Soft Finger pushing a press-button

The learned condensed FEM model of the Soft Finger robot with contact is evaluated on the action of pushing a press-button. The last one is modeled as a cube on which a constant known horizontal force T_{cube} opposed to the orientation of the bending movement is applied. The Soft Finger has to push this button following an horizontal trajectory. This trajectory is a set of positions of the center of the cube X_{cube} sampled along the aforementioned trajectory. As the finger bending is only possible in one direction, only a single degree of freedom of the cube is controlled. The results are shown in Figure 5.8.

On this application, the Soft Robot open-loop control using the condensed FEM model shows similar degree of precision with the simulation-based control.

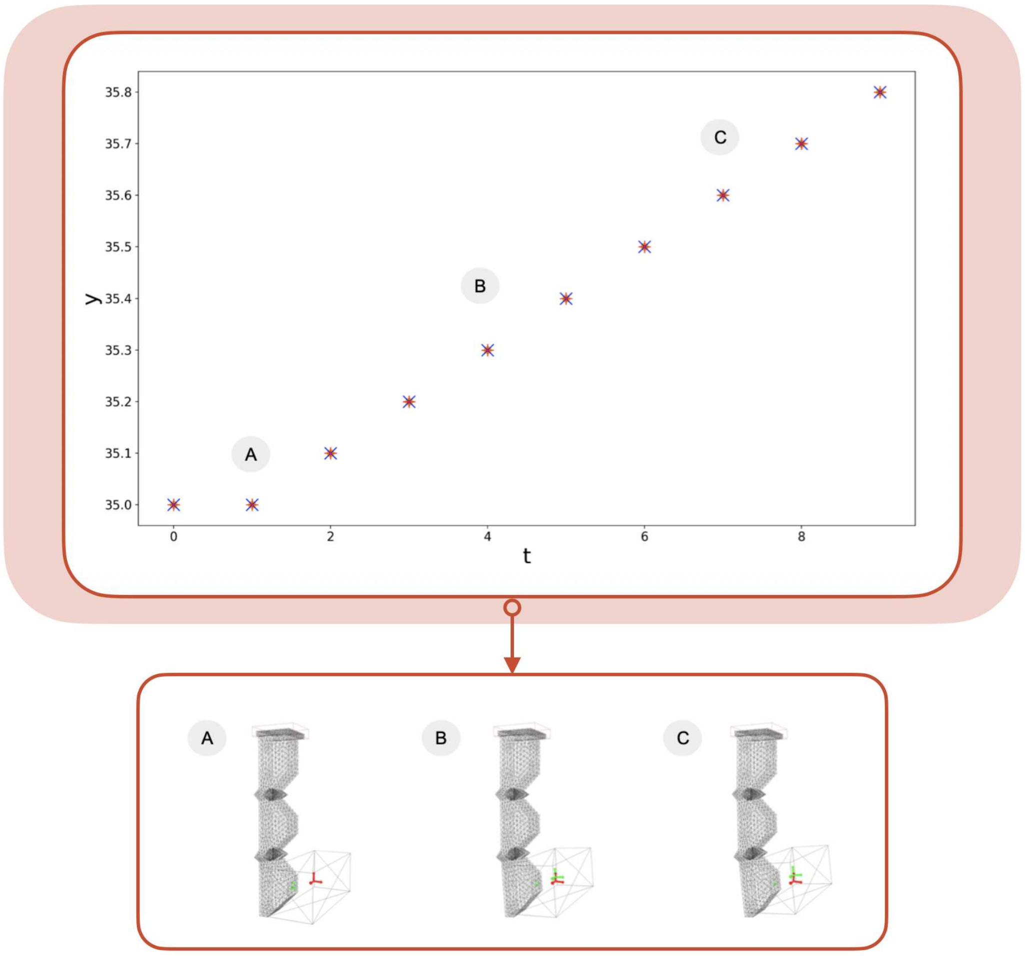


Fig. 5.8: Control results for the Soft Finger robot pushing a button. The reached horizontal position of the cube (mm) are compared on a trajectory of 10 successive goals (orange dots) when using learned mechanical matrices (blue cross) or mechanical matrices computed in simulation (red cross). 3 different positions of the Soft Finger robot met during this trajectory are shown as examples. On this representations, the effector is a green frame, and the goal a red frame.

Application: Soft Gripper Manipulating a Cube.

The mechanical matrices of the Soft Gripper robot are built from the mechanical matrices computed on three single Soft Finger. This shows that the condensed FEM model learned for a single robot can be used for predicting matrices of several similar robots in parallel. Compared to the scenario of the Soft Finger pushing a button, applied forces on the cube are not known this time.

After solving the same problem as for a single Soft Finger, Eq. 5.13 enables to express λ_c with the new values of λ_a and X_{cube}. The corresponding algorithm is written as follows:

Algorithm 1 Control algorithm for object manipulation scenario.

```
Start from $\lambda_c = 0$.
for each time step :
    1. Compute $\lambda_a$ and $X_{cube}$ according to $X_{cube} = \mathbb{A}\lambda_a + \mathbb{B}$ and the previous evaluation of $\lambda_c$. The value of $\lambda_c$ is used to calculate $\mathbb{B}$.
    2. Evaluate the new value of $\lambda_c$.
```

In this algorithm, λ_a and λ_c are updated once per time step. This is a simplifying assumption, which could lead to errors. However, it is not a major issue in the current scenario.

For the experiment, a circular trajectory of the object center is considered. The results are shown in Figure 5.9.

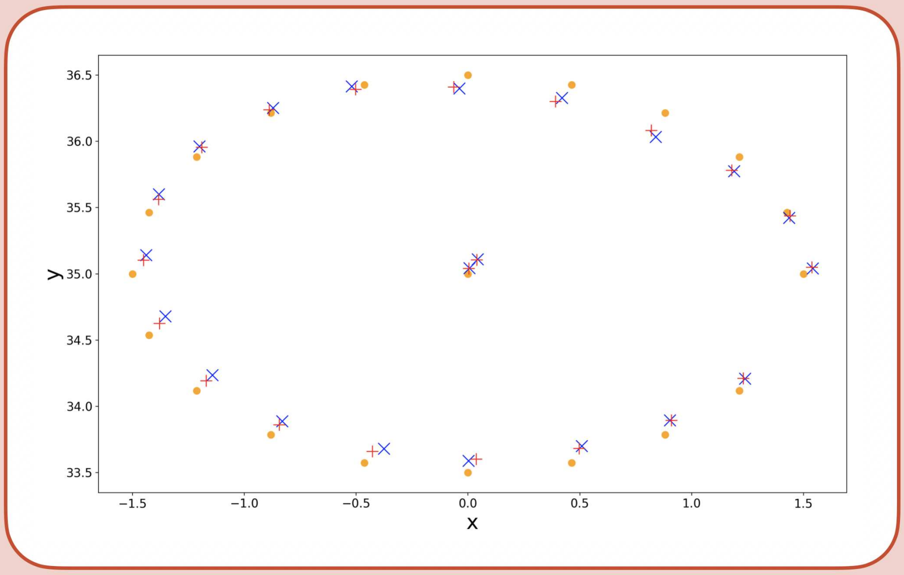


Fig. 5.9: Control results for the Soft Gripper robot. The position of the manipulated cube (mm) are compared on a trajectory for 20 different goals (orange dots) when using learned mechanical matrices (blue cross) or mechanical matrices computed from simulation (red cross).

On the considered trajectory, slight differences between goals and reached effector positions are observed both with the simulation-based and learning-based controllers. In addition, a small effector-goal gap of about 0.04 mm is observed along the vertical z-axes in both cases. This is explained by the modelling of the manipulator which considers Soft Fingers contact points to always be at the same location on the manipulated object. These errors could be reduced using a closed-loop controller.

Moreover, small differences between reached positions of both the simulation-based and learning-based controllers are observed. Compared to other cases from this paper, training results are less good, as can be seen in table 5.3. This is explained by the complexity of the dataset and may be improved through data filtering and hyper-parameters tuning.

5.1.5 Embedded control of soft robot using a condensed FEM modelling

The considered soft robot is a pneumatic trunk robot (see section 3.1.2) presented in [Cia+14] and called the Stiff-Flop robot. This robot is a highly flexible and pneumatically driven soft robot composed of two modules including three fully fiber-reinforced chamber pairs each. This robot was developed for medical applications, for example as an in-vivo cancer diagnostic tools during minimally invasive interventions [Cha+23]. The physical robot and the associated simulation are shown in Figure 5.10.

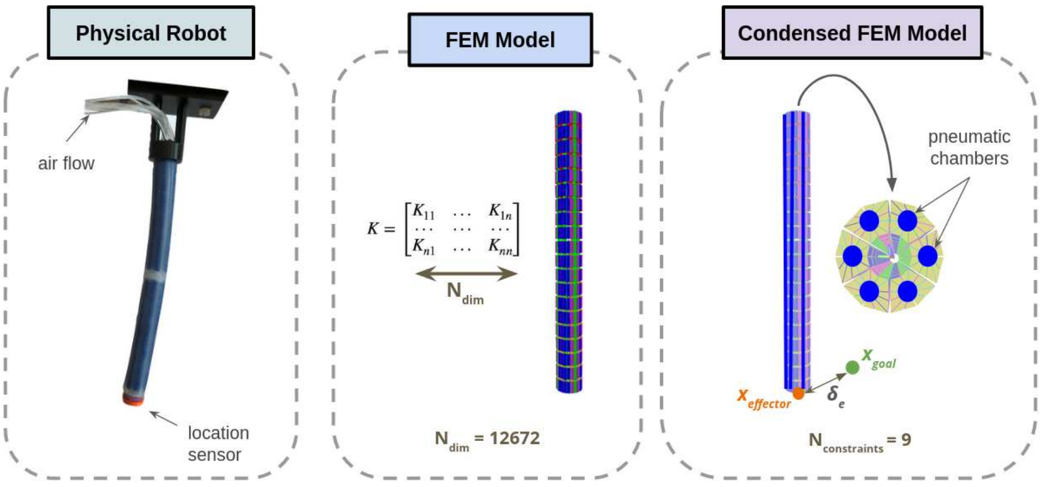


Fig. 5.10: Illustration of the robot used to illustrate embedded control. A) Stiff-Flop robot simulated in SOFA. The effector of the robot is represented by a green dot, and the goal by an orange dot. B) Physical prototype of the Stiff-Flop robot.

Concerning the simulation, the Stiff-Flop is built using a reduced finite element method model. The pneumatic forces are distributed all along the FEM nodes of the surface of the cavities. The distributed forces are projected on a reduced beam mathematical space using SOFA mapping, where the equilibrium of forces is solved. This reduced model compute faster than traditionnal FEM simulation, and enables a real time control of the robot.

For learning the condensed FEM model, data are collected in the space of the volume change of the cavities. Both the statistic about the sampling data and the condensed FEM learning results are show in Table 5.4.

Tab. 5.4: Data acquisition parameters for the Stiff-Flop robot for $n_l = 50000$ sample points. Actuator displacements are sampled in $[\delta_{a,min}, \delta_{a,max}]$ (mm^3). Final test loss values are displayed for 50000 learning iterations. The best trained network encountered is kept at best epoch.

Robot	Sampling bounds		Final Loss	Best Epoch
	$\delta_{a,min}$	$\delta_{a,max}$		
Stiff-Flop	0	30	9.22e-4	37070

Embedded open and closed-loop control using a condensed FEM model.

In this section, the framework for embedded open and closed-loop control of a soft robot based on the condensed FEM model is developed. The system is pressure-controlled and no additional actuator measurements are added. The condensed FEM model is used for computing the value of the volume increments based on the actuation λ_a and the projected mechanical matrices from Eq.2.14. This strategy results in a time step shift on the predicted mechanical matrices used for computing the actuation displacement. This shift may be corrected using a suitable corrector in a closed loop setting.

The general embedded control loop framework is as follows:

Algorithm 2 Embedded control loop.

```
Inputs: $W_0$ and $\delta_0^{free}$ from the training dataset.
Set initial actuation $\lambda_a = \mathbf{0}$, and initial distance $\delta_a = \delta_a^{free}$.

for each time step :
|    1. Recover the true position of the effectors from sensors.
|    2. If considering to close the loop, compute the corrected goal position.
|    3. Predict the mechanical quantities $\widetilde{W}$ and $\widetilde{\delta}^{free}$ using the trained neural
network.
|    4. Solve the optimization problem to recover the force $\lambda_a$.
|    5. Compute the new actuators state from mechanical matrices and forces as
$\delta_a = W_{aa}\lambda_a + \delta_a^{free}$.
|    6. Apply the forces $\lambda_a$ on the robot.
```

Closing the loop can be written using the δ_e^{free} value used for optimization. In our experiments, a PID controller is set from Eq. 5.6 as:

$$p_{goal,t}^{corrected} = p_{goal,t} + K \times \epsilon_t + I \times \sum_{t' \leq t} \epsilon_{t'} + D \times (\epsilon_t - \epsilon_{t-1}) \tag{5.14}$$

where $p_{goal,t}^{corrected}$ is the corrected goal at time t, $p_{goal,t}$ is the goal in the trajectory at time t, $\epsilon_t = p_{goal,t} - p_{effectors,t}$ is the position difference between the goal and the actual position of the effectors, K, I, D are the weight for the proportional, integral and derivative error. A unitary step is considered both for integral and derivative errors.

The full algorithm can be directly embedded on a microprocessor such as a Raspberry.

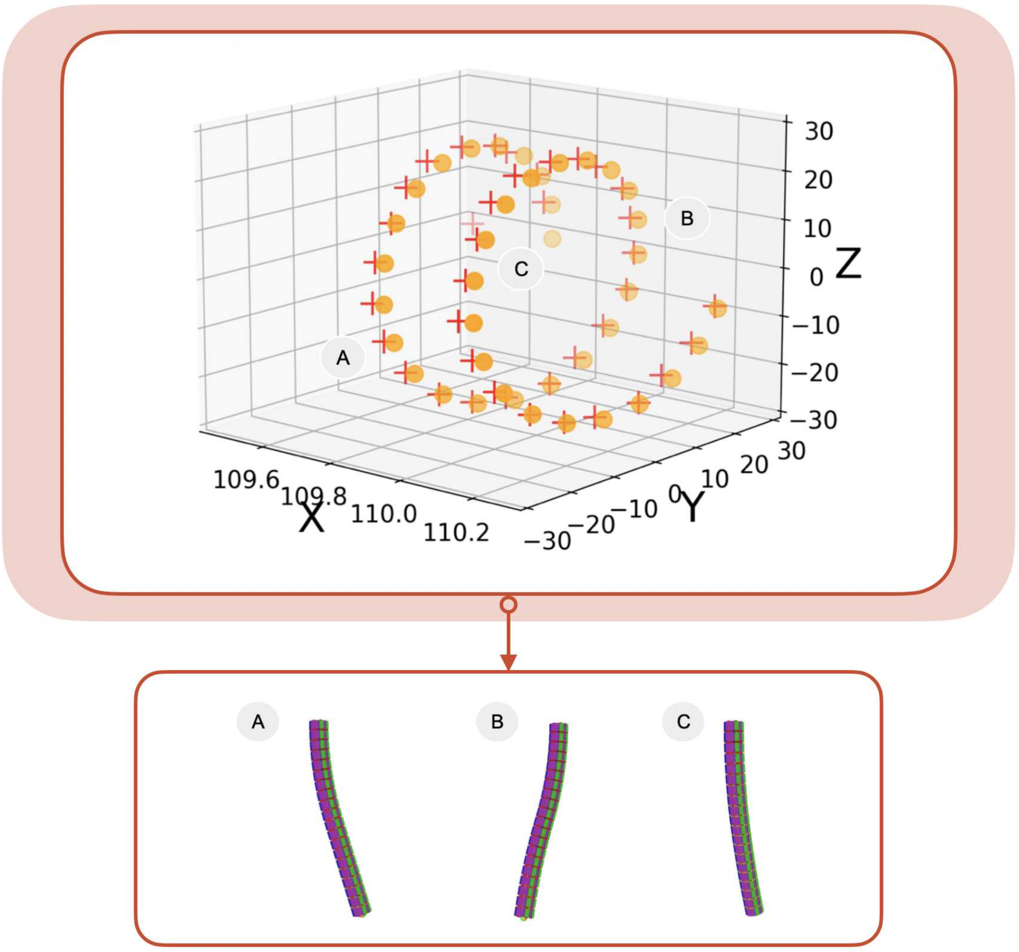


Fig. 5.11: Control results for the Stiff-Flop robot. The 3D-position (mm) of the end effector is compared on a trajectory for 40 different goals (orange dots) when using learned mechanical matrices (red cross). 3 different position of the Stiff-Flopp robot along this trajectory are shwon as examples. The effector is represented by a green dot, and the goal by an orange dot.

Experimental results with simulation in digital twin conditions.

The goal is to control the robot's physical prototype. To validate the learning, the robot simulation is used in digital twin conditions, to simulate the problem as if it was the physical prototype. The tests could not be carried out on the real prototype because of time constraints. The control is tested on the simulated robot and the results are shown in Figure 5.1.5. Both open and closed loop scenarios follow a spiral trajectory. There is a small z-offset between the goals and the actual positions of the end-effectors, but this offset is consistent with the different results found with other models. This error can be further reduced by fine-tuning the PID corrector. This validates the use of this model for robot control, under the same conditions as the physical prototype.

5.1.6 Condensed FEM modelling extended to calibration and design

In this section, we demonstrate how the condensed FEM model previously used to control soft robots is extended to handle both calibration and design optimization applications. Both the W and δ^{free} mechanical matrices directly capture the link between the robot's actuators, contact points and effectors. This link depends directly on the robot's body and its mechanical properties. Moreover, the function F linking both the W and δ^{free} mechanical matrices with the actuation state δ_a of the robot and the initial compliance matrix projected in the constraint space W_0 is fully differentiable. Differentiable fitness functions for assessing a design performance can then be designed from W and δ^{free}. As we will see, a model trained for several designs can be used to optimise that design without requiring FEM simulation. This is a critical point in design optimization loop when many designs have to be evaluated.

Many works in the community consider training a surrogate to directly learn the values of the fitness function relatively to the design variables. For instance, in [YHM23], a surrogate is learned for predicting angular deflections of pneumatic modules from both material properties and geometrical dimensions design parameters. Changing the fitness function then requires them to regenerate data and re-train a model. In the case of the condensed FEM model, since fitness functions are directly formulated from the learned matrices, a single condensed model can be used to optimize a robot across multiple tasks.

For this section, we use parametric design of a Soft Finger. This parametric design is lightly different than the one used in the previous sections and is an iteration of the one previously proposed in [Nav+23]. This Soft Finger is entirely made in silicone, contains two pneumatic cavities located at the joints and is actuated by one cable going from its base to the tip. This work showed how the Soft Finger geometry can be optimized both for its kinematics and the sensitivity of the pneumatic cavities which are used as a sensor, and also demonstrates the transfer to reality of the optimized results. Compared to this previous work, no pneumatic sensors and different design parameters are considered. The considered parametric Soft Finger is shown in Figure 5.12. Pulling on the cable has the effect of bending the finger and rigidify it. An end-effector is located on the fingertip to control its position.

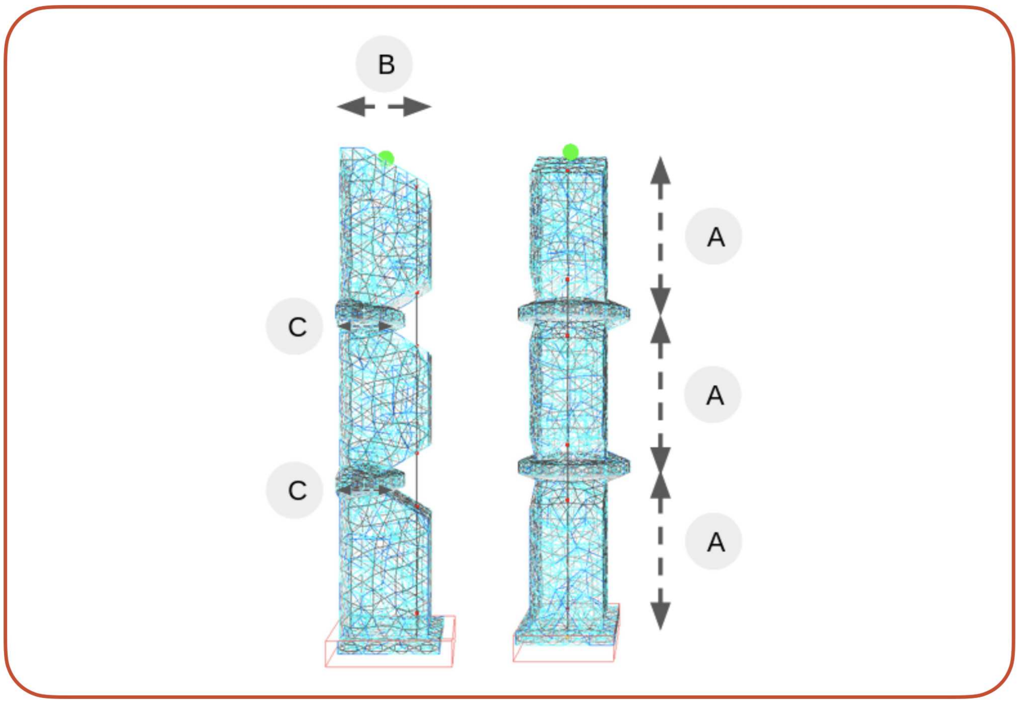


Fig. 5.12: Soft Finger parametric design. Design parameters are A) Length, B) Height and C) Joint Height.

Investigation on parametric design captured by a single condensed FEM model

In the following experiments, we show that the condensed FEM model is expressive enough to capture design variations. Two condensed FEM model are learned from two different dataset generated from the parametric Soft Finger. Both sampling strategies and learning results are introduced below:

- The **mechanical parameters dataset** considers variation of the Young's Modulus and Poisson Ratio mechanical parameters. Both displacement states and numerical simulation parameters are sampled using a Scrambled Halton sequence. A total of 10000 samples were simulated. Sampling parameters are described in Table 5.5.

Tab. 5.5: Bounds of sampling parameters for the condensed FEM model of the Soft Finger with varying mechanical parameters.

Parameter	Minimum Value	Maximum Value
Young Modulus (kPa)	1000	10000
Poisson Ratio	0.400	0.499

- The **geometrical dimensions dataset** considers geometrical dimensions variations around a baseline design of the Soft Finger. The three geometrical design parameters are described in Figure 5.12. A total of 50000 samples were simulated. Sampling parameters are described in Table 5.6.

Tab. 5.6: Bounds of sampling parameters for the condensed FEM model of the Soft Finger with varying geometrical dimensions.

Parameter	Minimum Value	Maximum Value
Length (mm)	38	42
Height (mm)	20	22
Joint Height (mm)	5.0	8.0

Training results are displayed in Table 5.7 for both datasets. For learning the condensed FEM model with geometrical variations of the Soft Finger, a more complex strategy is adopted. A neural network of 5 fully connected layers with 800 nodes each is chosen. As the design space is exponential in the number of parameters and actuations, and under-sampled in the training set, a dropout strategy is applied for regularizing the network. Practically, each hidden layer node has a probability of 0.4 of being disregarded at each epoch of the training. The orders of magnitude of the final loss for the geometrical dimensions dataset displayed in the Table is then much higher than for the usual losses, but the generalization is sufficient enough regarding targeted applications. Taking a closer look into element-wise prediction errors, mechanical matrices values related to the link between actuation and effectors in the direction where the geometrical dimensions (Length and Height) of the parametric Soft Finger evolve are harder to learn. However this standardized loss error values corresponds to a small difference between predicted and simulated effector positions of around 3mm in the worst case met on the entire test dataset. These results can be improved by increasing the size of the dataset or by learning conjointly the two networks F and G.

Tab. 5.7: Test loss values for each parametric Soft Finger dataset for both G and F neural networks.

Dataset	G		F	
	Final Loss	Best Epoch	Final Loss	Best Epoch
Mechanical Parameters	3.82e-7	18420	5.04e-5	12530
Geometrical Dimensions	0.19	4400	0.13	270

For showing that the condensed FEM model provides a good characterization of the design, several designs are evaluated on a trajectory using the condensed FEM model based open loop control. The considered trajectory is the one showed in Figure 5.4 (closing the finger). Results are desplayed as errors between positions reached by the ground truth simulation and the corresponding location obtained with the condensed FEM model. They are displayed in Table 5.8. Whatever the sampled design, it shows that a trained condensed FEM model on the proper dataset enables to control different designs from this dataset. On studied examples, maximal positioning error is about 1.2 mm which is approximately less than 1% of the total length of the parametric Soft Finger.

Tab. 5.8: Positioning error for different values of geometrical and mechanical parameters. Error is defined as $\frac{|d_{simu}-d_{learned}|}{d_{simu}}$, where d_i corresponds to the mean euclidean distance between the effector and the goal evaluated over the trajectory, and obtained whether from simulation or condensed FEM model.

Trajectory errors with mechanical parameters dataset:

Young Modulus (GPa)	Poisson Ratio	Error
5000	0.47	0.91%
1000	0.45	0.29%
7000	0.4	0.90%
3000	0.47	1.03%

Trajectory errors with geometrical dimensions dataset:

Length (mm)	Height (mm)	Joint Height (mm)	Error
40	20	6	1.03%
38.5	21.5	6	3.11%
40	20.5	7.5	7.02%
41	21	5.5	1.62%
39.5	20	7	5.10%

Application to calibration of the condensed FEM model of a parametric Soft Finger

Numerical parameters, such as the Young's Modulus and the Poisson Ratio, influence the behavior of the robot in simulation. Since the condensed FEM model is learned from simulations, one of the drawbacks is that it is learned for a fixed set of numerical parameters. Even if the simulation is calibrated beforehand on a physical prototype, defaults may appear during the manufacturing of other similar physical prototypes or due to silicone degradation overtime, and it would then be necessary to re-calibrate the simulation and learn a new condensed FEM model for controlling them. In the example of the Soft Finger, the appearance of air bubbles or an incorrect dosage of silicone during the molding process can lead to different corresponding mechanical parameters. For dealing with this issue, the extended condensed FEM model is learned with variations of the numerical parameters of the soft robot design. The aim is to address learning the condensed FEM model with zero-shot transfer to reality.

Once the condensed FEM model is learned, it is calibrated using data collected from the physical robot. The general workflow to calibrate a condensed FEM model of any soft robot is as follows. First, positions of the effector(s) are measured for several actuation displacement states δ_a. Increasing the number of actuation states enables better convergence. Then, a fitness function $O_{cal}(P)$ depending of the design parameters P and computing the difference between the relative position of the effectors δ_e^* on the physical prototype and those estimated by the condensed FEM model $\widetilde{\delta}_e$ is built as follows:

$$O_{cal}(P) = \sum_{\delta_a} |\widetilde{\delta}_e(\delta_a, P) - \delta_e^*(\delta_a)| \tag{5.15}$$

In the previous equation, dependence of the different quantities to both P and δ_a are explicitly shown for clarity.

In order to build a differentiable link with the design parameters P, δ_e is expressed from the predicted mechanical matrices. Using Eq. 2.14 without contacts, it gives:

$$\widetilde{\delta}_e(\delta_a, P) = \widetilde{W}_{ea}(\delta_a, P)\widetilde{W}_{aa}^{-1}(\delta_a, P)(\delta_a - \widetilde{\delta}_a^{free}) + \widetilde{\delta}_e^{free} \tag{5.16}$$

The design parameters P are optimized through gradient descent through minimizing O_{cal}.

Back to the Soft Finger example, the position of the effector on the tip is easily observable with a camera, which makes it a good candidate as a calibration data. Moreover, only the Poisson Ratio is optimized as the Soft Finger is controlled in displacement, the Young's Modulus therefore being not a relevant design parameter for the calibration. The experimental validation of the calibration of the learned condensed FEM model of the Soft Finger is explained below. As its simulation to reality transfer has already been analysed in previous work [Nav+23], the states δ_e^* of the robot are sampled in simulation for cable displacement values δ_a of 0mm, 5mm and 10mm. For the moment the method has not yet been tested with a physical robot. Young's Modulus and Poisson Ratio ground truth parameters are chosen equal respectively to 3000 GPa and 0.47. Then, the fitness function from Eq. 5.15 is minimized as previously introduced for retrieving the corresponding parameters. Obtained convergence history are displayed in Figure 5.13 for three different initializations of the mechanical parameters.

In each case, the algorithm succeeds in finding the good Poisson Ratio for reducing the distance between the positions of the effector given by the learned model and those of the physical robot. As similar final fitness functions are obtained, it shows that the algorithm converges regardless of the mechanical parameters

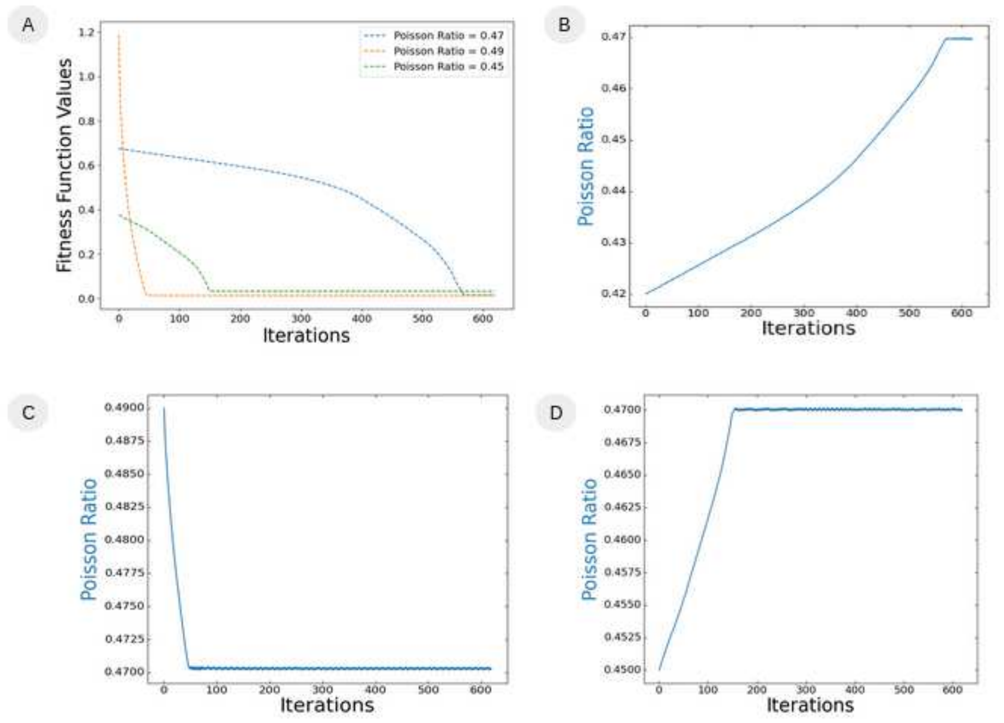


Fig. 5.13: Optimization history for the calibration of the Soft Finger robot. Results have been obtained with an initial learning rate of 0.01 used conjointly with an adaptive scheme as described in section 5.1.2. Both fitness function (A) and mechanical parameters histories (B,C,D) across optimization iterations are displayed for different initial guesses of the mechanical parameters. Initial mechanical parameters are a fixed Young Modulus of 3000 GPa as well as B) Poisson Ratio = 0.47, C) Poisson Ratio = 0.49 and D) Poisson Ratio = 0.45.

initialization on this scenario. Here the Young's modulus has no impact on the fitness function from Eq. 5.15 because effector displacements are evaluated for different cable displacements. Changing for a servoing in force exerted on the cables instead of cable displacements would require to take into account the Young's Modulus during the optimization. With this experiment, it is demonstrated that it is possible to learn a single condensed FEM model that can be directly calibrated to a soft robot. By relying on the differentiability of the learned condensed FEM model, the calibration method only takes a few seconds on a pre-learned model, compared to other models relying on non-gradient solvers costing a lot of simulations.

Application to design optimization of a parametric Soft Finger

In this subsection, we demonstrate how the condensed FEM model could be used for optimizing the design of a soft robot on the case of the parametric Soft Finger. Alongside the intended design optimization application, including design parameters such as geometrical dimensions makes possible to calibrate the condensed FEM model to design variations which may appear during the manufacturing process, using similar technique as the one shown in the previous section.

For this experiment, the parametric Soft Finger design introduced in Figure 5.12 is optimized successively for its dexterity and for precision grasping. Design optimization objectives are formulated from the predicted mechanical matrices. The two considered objectives are both illustrated in Figure 5.14.

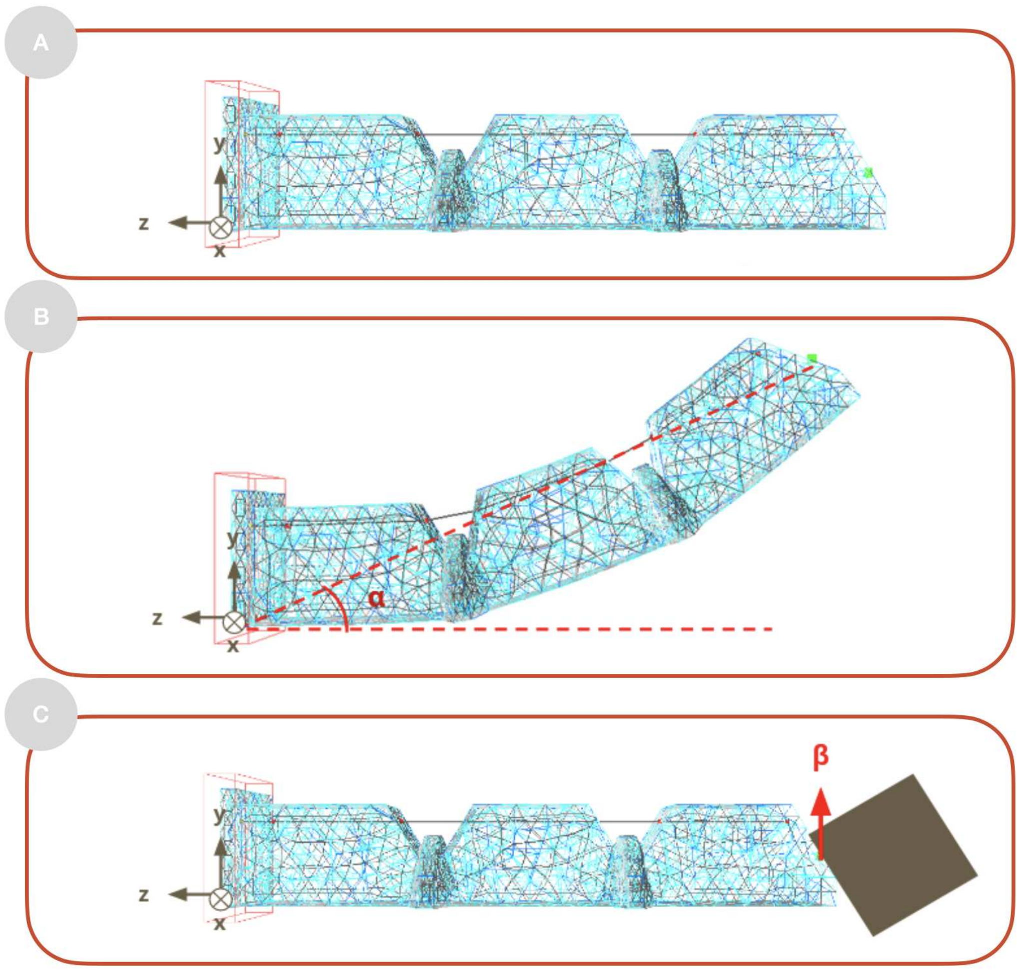


Fig. 5.14: Illustration of design objectives for the parametric Soft Finger. A) Soft Finger in the initial state. B) Bending angle $\alpha(\delta_a, p)$ reached for a fixed cable displacement δ_a. C) Contact force $\beta(\delta_a, p)$ generated for a fixed cable displacement δ_a.

The first fitness function O_{dext} characterizes the kinematics of the parametric Soft Finger, that is the reached bending angle for a fixed cable displacement δ_a of

10mm. Minimizing this function is equivalent to maximizing the workspace of the robot. O_{dext} is expressed as follows:

$$\begin{aligned} O_{dext}(P) &= |\alpha^{max} - \alpha(\delta_a, P)| \\ \alpha(\delta_a, P) &= Arccos(\frac{\tilde{\delta}_e^z(\delta_a, P)}{|\delta_e(\mathbf{0}, P)|}) \end{aligned} \tag{5.17}$$

where α is the function giving the bending angle of the parametric Soft Finger in the z-plane and $\alpha^{max} = 1.57$ rad is the maximum reached angle.

The second fitness function O_{str} measures the contact force generated at the tip of the Soft Finger for a fixed cable displacement δ_a of 10mm. Minimizing this function amounts to maximizing using the parametric Soft Finger for precision grasping. Again, for building a differentiable link with the design parameters P, the contact force λ_c is expressed from the predicted mechanical matrices. If we assume that the object is not moving ($\delta_c = 0$) and that the tip of the end-effector is a contact point, we can use the expressions for δ_a and δ_c in Eq 2.14 to write:

$$\lambda_c(P, \delta_a) = B(P, \delta_a)^{-1}[\tilde{W}_{ca}\tilde{W}_{aa}^{-1}(\delta_a - \tilde{\delta}_a^{free}) + \tilde{\delta}_c^{free}] \tag{5.18}$$

with $B(P, \delta_a) = \tilde{W}_{ca}\tilde{W}_{aa}^{-1}\tilde{W}_{ac} - \tilde{W}_{cc}$ Finally, the fitness function O_{str} is expressed as follows:

$$\begin{aligned} O_{str}(P) &= |\beta^{max} - \beta(\delta_a, P)| \\ \beta(\delta_a, P) &= |\lambda_c(P, \delta_a)^y| \end{aligned} \tag{5.19}$$

where β is the function giving the force generated at the tip of the parametric Soft Finger on the y-axis and $\beta^{max} = 10000$ N is an unreachable force.

Single objective optimization:

As the considered fitness functions are highly non convex, the simple gradient descent optimizer used may fall in a local optima. For avoiding this behaviour, a grid search is first performed to determine a good starting point for the optimization. As the design space is of only three dimensions, the computation is a simple matter of seconds. A total of 600 designs are evaluated for both fitness functions. On high dimensional cases, an alternative for avoiding the dependency to the initialization would have been to use stochastic gradient descent instead of a simple gradient descent scheme as a solver.

For each fitness function, the objective function landscape is computed. For this purpose, a regular grid sampling strategy is applied around the chosen initialization starter design. Landscapes computed for both fitness functions are displayed in Figure 5.15. They show that the length has a higher impact on both objectives than two other design variables.

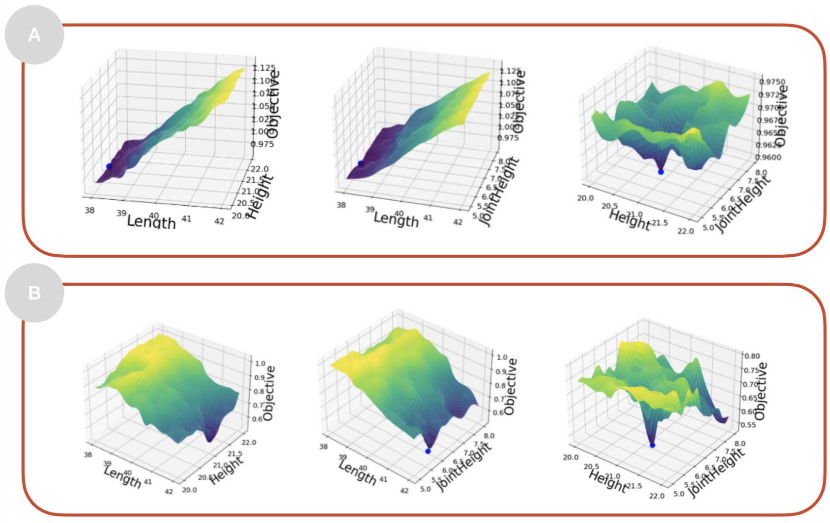


Fig. 5.15: Fitness landscapes around the initialized design selected with grid search strategy, for A) dexterity fitness function (Length = 38.0mm, Height = 20.57mm, Joint Height = 6.28mm) and B) strength fitness function (Length = 42mm, Height = 22mm, Joint Height = 8mm). The best design after optimization is shown as a blue dot.

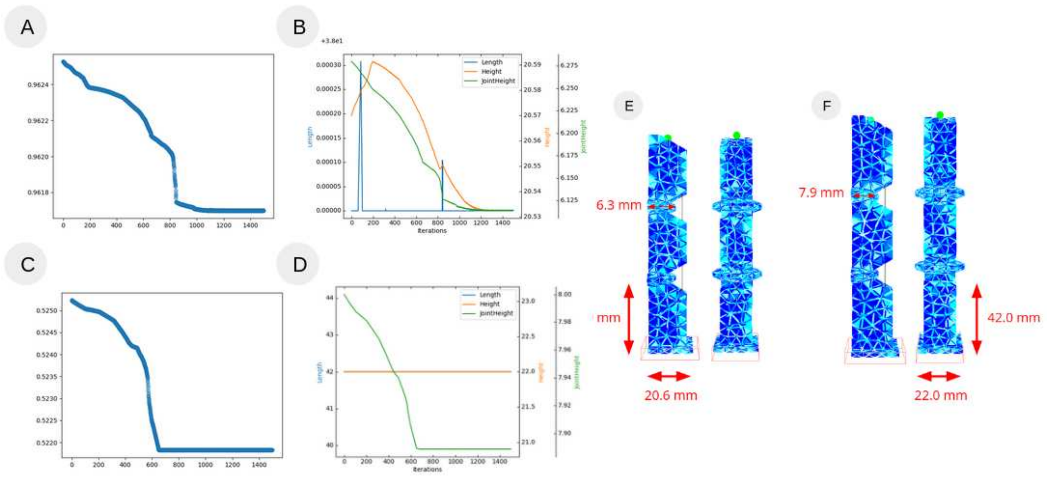


Fig. 5.16: Optimization histories and results for single objectives case on the parametric Soft Finger. Fitness functions histories respectively for A) kinematics and C) strength. Design parameters histories respectively for B) kinematics and D) strength. Resulting designs respectively for E) kinematics and F) strength.

Optimization history and final results are shown in Figure 5.16. Regarding the dexterity, the best design privileged a shorter length whereas longer length helps to generate more forces on the object thanks to the lever arm effect. The best design regarding contact forces generated at the tip is also a Soft Finger with more material, enabling to rigidify better around the contact location.

Multi-objective optimization:

In this experiment, it is highlighted how to determine the optimal design of a Soft Finger regarding several metrics computed using the condensed FEM model. Figure 5.17 shows the fitness functions values for 600 Soft Finger designs sampled with a grid search strategy. For having comparative values between the two losses, they are both normalized using $O_i^{norm}(P) = \frac{O_i(P) - min_{P*}(O_i(P*))}{max_{P*}(O_i(P*)) - min_{P*}(O_i(P*))}$ where $P*$ are design parameters encountered during initial grid search.

A Pareto front is clearly visible, showing that the two considered metrics are antagonistic. A compromise has therefore to be found. This can be done using the gradients calculated via the condensed model, through formulating a single fitness function as an aggregation of the considered fitness functions:

$$O_{mult}(P) = \gamma_1 O_{dext}^{norm}(P) + \gamma_2 O_{str}^{norm}(P) \tag{5.20}$$

where both γ_1 and γ_1 are user-chosen weighting factors between 0 and 1 for selecting the priority to give to each fitness functions. This kind of approach enables to generate the optimal design of a Soft Robot given user-chosen requirements.

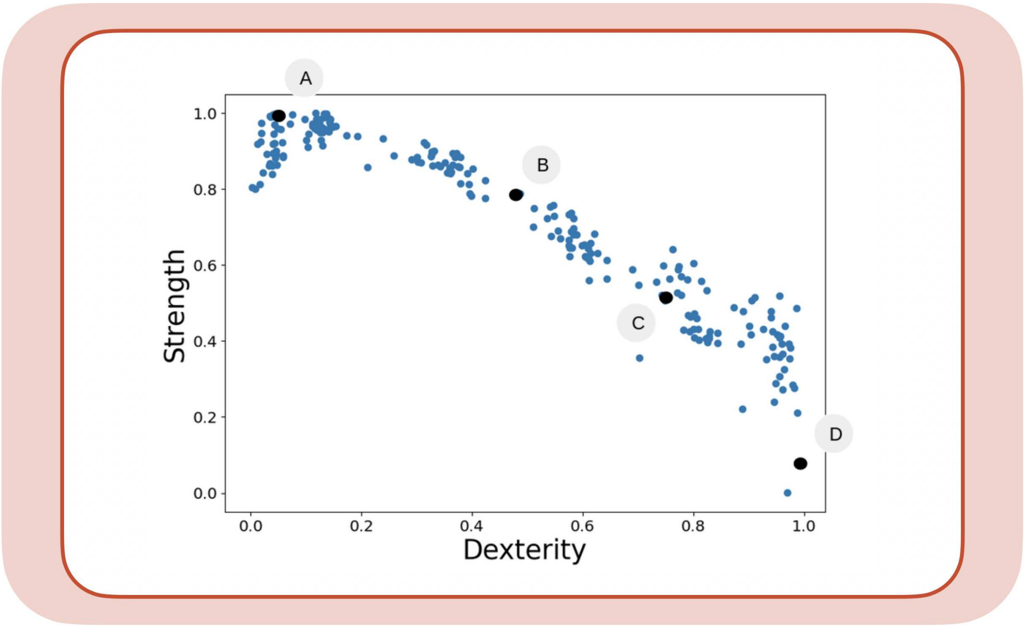


Fig. 5.17: Pareto Front for the design optimization of the Soft Finger regarding Dexterity and Strength metrics and generated using a Grid Search strategy. Design parameters for several design sampled: A) Length = 38.0mm, Height = 21.6mm, Joint Height = 5.0mm, B) Length = 40.4mm, Height = 20.0mm, Joint Height = 5.6mm, C) Length = 40.2mm, Height = 22.0mm, Joint Height = 7.4mm, D) Length = 42.0mm, Height = 22.0mm, Joint Height = 8.0mm

5.1.7 Discussion

In this section, we have demonstrated the potential of learning a condensed FEM model of the robot. First, we have shown that it can be use for the control of the robot itself, in particular for direct or inverse kinematics. Then we have extended the formulation to contact. Contacts are considered as constraints in the same way as actuators and effectors. We show how to formulate inverse control problems both for a single and coupled soft robots with contact modelling. In the current framework, both the number and the location of the contact points on the robot are fixed. This is an important limitation as it does not allow for unanticipated obstacles. This limitation is due to the data input and output size dependency of the MLP used for learning the condensed FEM model. As a workaround, size-independent neural networks such as Transformers [Vas+17] may be integrated in the framework.

Secondly, tools for embedding control have been explored. We proposed an open-loop and closed-loop control based on the learned condensed FEM model of a digital twin conditions simulation. Future work has to be carried out to control a physical prototype directly. Integration into a low-capacity controllers is possible thanks to the high motor command generation frequency (1e3Hz for the considered robot) and the low memory consumption of the condensed FEM model. The embedded control could be applied on other robots. The more complex is the robot, the more interesting the time gain is. So far, only kinematics has been considered, and the condensed FEM model extension to dynamics may unlock applications in using more robust controllers such as Model Predictive Control.

Finally, we show that a single condensed FEM model is able to simultaneously capture design and actuation states variations. Thanks to the high prediction speed and the inherent differentiability of the model, applications in direct calibration of the condensed FEM model and design optimization of soft robots are demonstrated on the example of a parametric Soft Finger. This could open a new path to design and control co-optimization applications based on a design-dependant surrogate.

Although we consider entirely soft robots, the current condensed formalism enables to consider hybrid robots with both rigid and soft parts.

5.2 Condensed dynamics for optimal control in soft robotics

Inspired by the one-step control in soft robotics and the trajectory optimization tools developed in rigid robotics (see Section 2.2.3), we propose a formulation of the trajectory optimization problem in soft robotics based on the elaboration of a condensed FEM dynamics. The general idea of the method is to use the complete FEM dynamics of the soft robot to simulate its behavior. Then a backward step uses the linearized condensed FEM dynamics to find the best actuation to solve the given task.

5.2.1 Condensed dynamics for trajectory optimisation

Condensed dynamics of soft robots

In the multi-step control problem, we have to write the dynamic of the system. In Chapter 2, we presented the choice of solving the general equation of dynamics using a semi-implicit Euler scheme. At each time step, the system presented in (2.4) and reproduced here has to be solved:

$$A d\dot{q} = h f_{int}(q_{k-1}, \dot{q}_{k-1}) + h^2 K \dot{q}_{k-1} + h f_{ext} \tag{5.21}$$

This scheme is semi-implicit because the internal forces are linearised around the current position, with only one linearisation per time step. Considering this equation and making the value of actuation explicit, we can write the following system:

$$d\dot{q} = A^{-1} b(q_{k-1}, \dot{q}_{k-1}) + h A^{-1} H_a^T u_{k-1} \tag{5.22}$$

where $b(q_{i-1}, \dot{q}_{i-1}) = h f(q_{i-1}, \dot{q}_{i-1}) + h^2 K \dot{q}_{i-1} + h f_{ext}$. Using the relation $p = H_e q$ between the position of the effectors p and the position of all the point of the mesh q, we can project the dynamics in the effector space according to:

$$d\dot{p} = H_e A^{-1} b(q_{k-1}, \dot{q}_{k-1}) + h H_e A^{-1} H_a^T u_{k-1} \tag{5.23}$$

where we recognize the condensed FEM matrices W in the right member of the equation. Using the implicit integration scheme, we can write:

$$\begin{aligned} \dot{p}_k &= \dot{p}_{k-1} + H_e A^{-1} b(q_{k-1}, \dot{q}_{k-1}) + h W_{ea} u_{k-1} \\ p_k &= p_{k-1} + \dot{p}_k h = p_{k-1} + h\dot{p}_{k-1} + h H_e A^{-1} b(q_{k-1}, \dot{q}_{k-1}) + h^2 W_{ea} u_{k-1} \end{aligned} \tag{5.24}$$

The condensed dynamics of the soft robot can be written in matrix notation:

$$\begin{aligned} X_{k+1} &= \begin{bmatrix} h^2 W_{ea,k} \\ h W_{ea,k} \end{bmatrix} u_k + \begin{bmatrix} I & hI \\ 0 & I \end{bmatrix} X_k + \begin{bmatrix} hI \\ I \end{bmatrix} H_e A^{-1} b(q_k, \dot{q}_k) \\ \Leftrightarrow X_{k+1} &= f_k(X_k, u_k) \end{aligned} \tag{5.25}$$

where $X_k = [p_k, \dot{p}_k]^T$

Derivative of the condensed dynamic

As we explain in section 2.2.3, DDP uses the derivative of the dynamics to compute the Hamiltonian value. By derivating Eq. 5.25 according to u_k and X_k we have for the control part:

$$\frac{\partial f_k}{\partial u} = \begin{bmatrix} h^2 W_{ea,k} \\ h W_{ea,k} \end{bmatrix} \tag{5.26}$$

and for the state using the chain rules and the fact that H_e is its own pseudo-inverse:

$$\frac{\partial f_k}{\partial X} = \underbrace{\begin{bmatrix} I & hI \\ 0 & I \end{bmatrix}}_{\frac{\partial f_k}{\partial X}_{linear}} + \underbrace{\begin{bmatrix} H_e A^{-1} & 0 \\ 0 & H_e A^{-1} \end{bmatrix} \begin{bmatrix} h^2 K & h^3 K + h^2 D \\ hK & h^2 K + hD \end{bmatrix} \begin{bmatrix} H_e^T & 0 \\ 0 & H_e^T \end{bmatrix}}_{\frac{\partial f_k}{\partial X}_{non-linear}} \tag{5.27}$$

This quantity is composed of two termes, one corresponding to the derivative of the linear part of the dynamics, and the other to the derivative of the non-linear part.

Constraint on the actuation

We can define a mathematical constraint $h_k(X, u) \leq 0$ that corresponds to the limits of the actuation values. This constraint can be expressed as:

$$h_k(X, u) = h_k(u) = \begin{bmatrix} -I \\ I \end{bmatrix} u + \begin{bmatrix} u_{min} \\ -u_{max} \end{bmatrix} \leq 0 \tag{5.28}$$

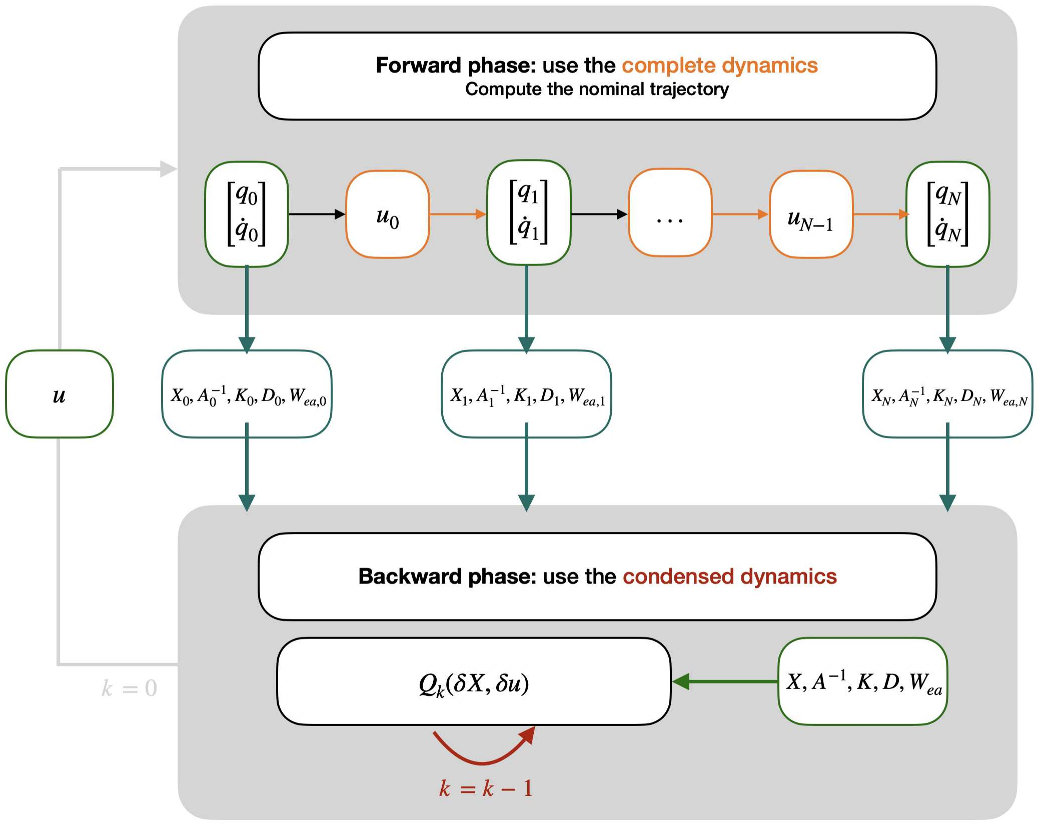

Forward phase: use the complete dynamics
Compute the nominal trajectory
$\begin{bmatrix} q_0 \\ \dot{q}_0 \end{bmatrix}$
u_0
$\begin{bmatrix} q_1 \\ \dot{q}_1 \end{bmatrix}$
$\dots$
u_{N-1}
$\begin{bmatrix} q_N \\ \dot{q}_N \end{bmatrix}$
u
$X_0, A_0^{-1}, K_0, D_0, W_{ea,0}$
$X_1, A_1^{-1}, K_1, D_1, W_{ea,1}$
$X_N, A_N^{-1}, K_N, D_N, W_{ea,N}$
Backward phase: use the condensed dynamics
$Q_k(\delta X, \delta u)$
X, A^{-1}, K, D, W_{ea}
$k = 0$
$k = k - 1$

Fig. 5.18: Illustration of the trajectory optimization pipeline. Starting from an initial position, the forward phase computes the different positions and velocities of the complete mesh using the full dynamics (in orange). From the simulation data, a nominal trajectory is extracted and the mechanical quantities necessary for the DDP are saved (in blue). The nominal trajectory (green) is used with the condensed trajectory (red) to compute the new action sequence that will be given to the forward pass. Several forward and backward steps may be necessary to converge to a solution.

General method to solve the trajectory optimisation problem

The derivatives used for the optimization problem depend on the mechanical quantities W_{ea}, A^{-1}, K and D. Technically these quantities depend on the position and velocity of all the points of the mesh of the robot. We assume that we are looking for an actuation around a nominal trajectory and that the mathematical quantities vary slightly around this trajectory. This means that these quantities are constant for one time step of the backward phase.

It is therefore necessary to compute a faithful nominal trajectory to obtain useful admittance, stiffness and damping values for the optimisation problem. To do so, the forward phase uses the complete dynamics of the robot and the simulation is used to compute the next values of W_{ea}, A^{-1}, K and D for the backward phase. When only the derivative of the linear part of the dynamics is used in the backward phase, only W_{ea} is recovered from the simulation. The global method is summarized in Figure 5.18 (see Section 2.2.3 for notations).

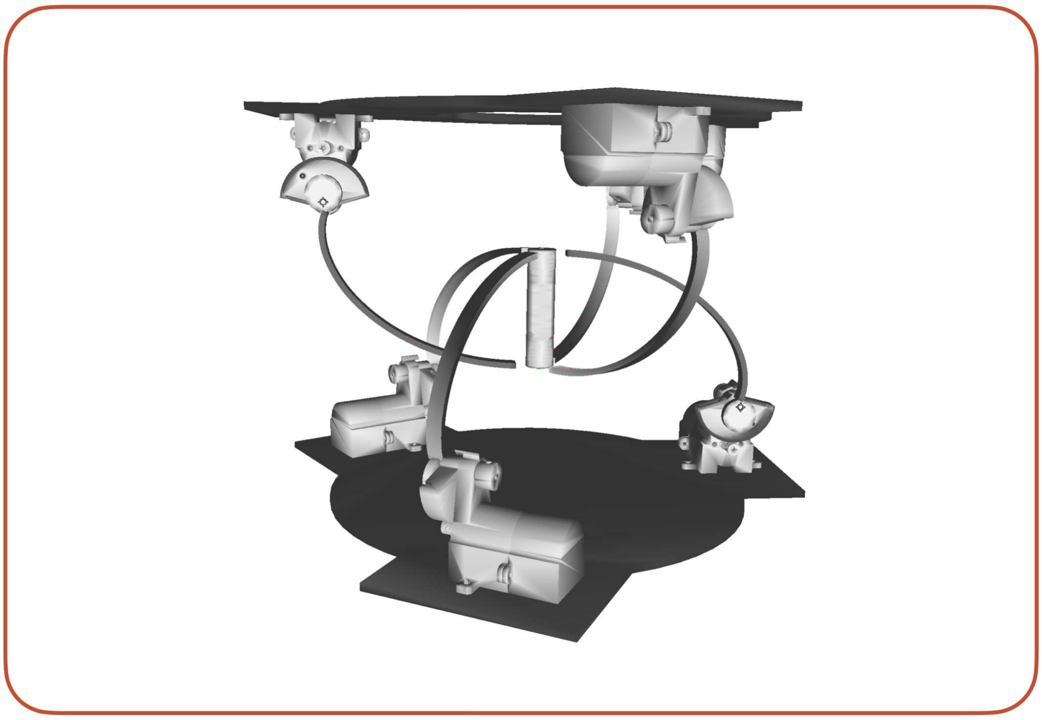


Fig. 5.19: Simulation of the Lianford robot.

5.2.2 Trajectory optimisation

We propose to illustrate the method using three different robots: the Diamond robot (see section 3.1.2), the Stiff-Flop robot (see section 5.1.5) and the Lianford robot. The Lianford robot is a soft parallel robot with 6 deformable legs developed for haptic applications. Each end of the legs is attached to a DC motor, whose rotation causes the legs to deform. The effector is attached to a central handle, whose movement is imposed by the deformation of the legs. This robot is illustrated in Figure 5.19. The trajectory optimization is performed using the proxddp solver [Jal+22]. This solver relies on a primal-dual augmented Lagangian approach to solve the optimization problem. The user interface allows to define the underlying optimal control problem on a node-by-node basis, to write the dynamics and constraints, and to specify the differences between forward and backward stages. The forward step has been adapted in order to make the link with SOFA. The results are presented for a fixed maximum number of iterations and tolerance.

5.2.3 Scenario 1: Use both derivative of the linear and non linear part of the dynamic in the backward step.

The first scenario compares the positioning performance obtained when the derivative of the non-linear part of the dynamics is used in the backward phase, and when this is not the case. This comparison is illustrated with the Diamond robot and shown in Figure 5.20. The goal is to position the effector of the Diamond at two successive positions.

From these results we can see that the use of condensed dynamics allows to control the position and velocity of the robot, both with and without the derivative of the non-linear part of the condensed dynamics. The results in terms of positioning error are similar with and without the use of $\frac{\partial f_i}{\partial X}_{non-linear}$. However, the number of iterations to reach the same performances is more important with the non linear part (reach the maximum of 1000 iterations set for this experiments) rather than without (208 iterations). In addition, using the derivative of the non-linear part leads to additional calculations, as the resulting matrix is not calculated directly in the forward stage. More precisely, the matrix W is obtained with $O(m^2)$ operations, where m is the number of constraints. The non-linear part of the derivative is obtained with $O(m \times n)$ operations, where n is the number of elements in the FEM mesh. n is a large number that depends on the accuracy of the FEM model. Therefore, considering only the linear part of the derivative of the dynamics according to X in the backward stage improves the calculation time.

The use of the derivative of the linear part alone in the backward is sufficient to control the soft robot. This is because the full non-linear FEM model is used to find the nominal trajectory, and linearisation around this nominal trajectory in the backward is sufficient to converge to a solution. In the remainder of this work, only the derivative of the linear part of the dynamics is used.

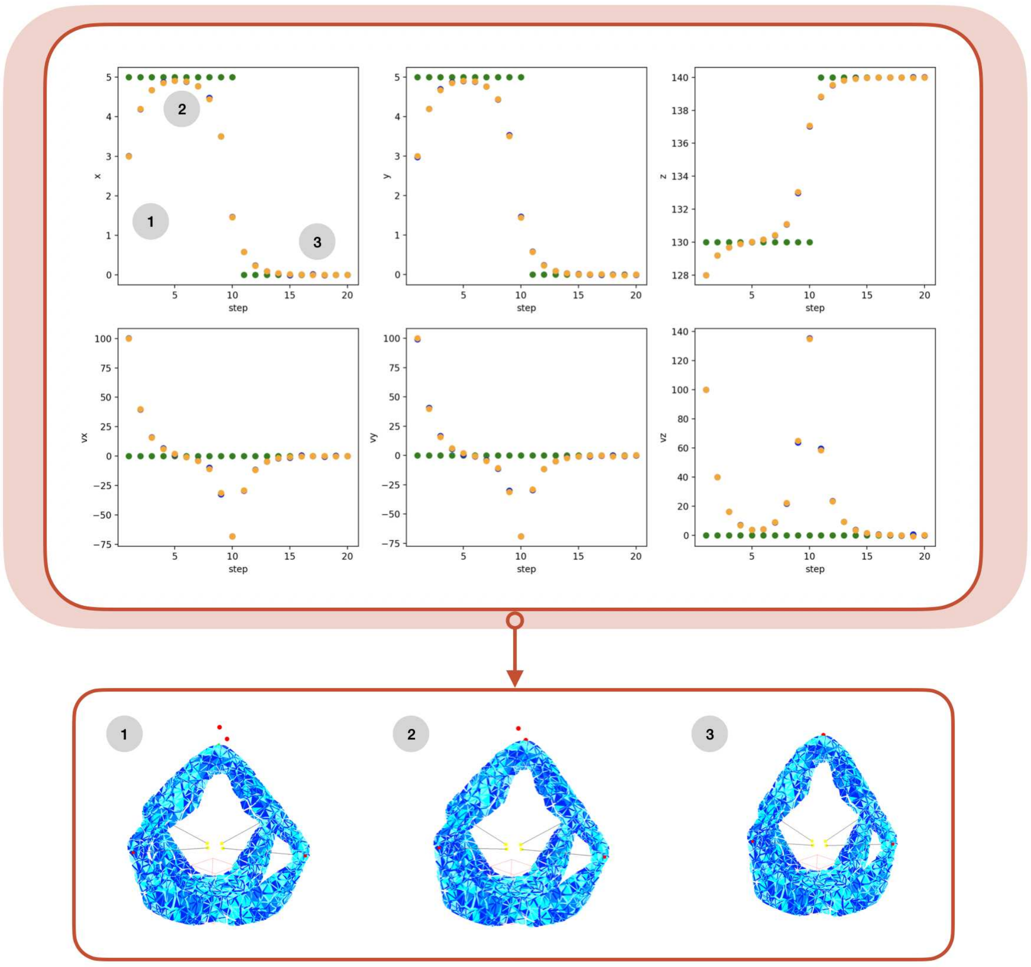


Fig. 5.20: Results obtained for Diamond robot control. Top, from left to right: end effector position along x, y and z axes (mm). Middle, left to right: end effector speed along x, y and z axes (mm/s). Green: the objective. Blue: results obtained by considering the derivative of the non-linear part of the condensed dynamics. Orange: results obtained by considering only the linear part. Three robot positions along the trajectory (1, 2, 3) are shown.

5.2.4 Scenario 2: Constraint actuation with the PneumaticTrunk.

The second scenario shows that we can constraint the actuation during the trajectory. This scenario is illustrated using the PneumaticTrunk. The results are shown in Figure 5.21. This robot illustrates the method with a different type of actuation. The objective is to position the end-effector attached to the centre or the end of the robot.

The results are shown for one axis and one actuators, but similar results can be plot for the other axis and actuators. The framework enable to take into account bounds on the actuation, and can integrate different kind of geometry and actuation.

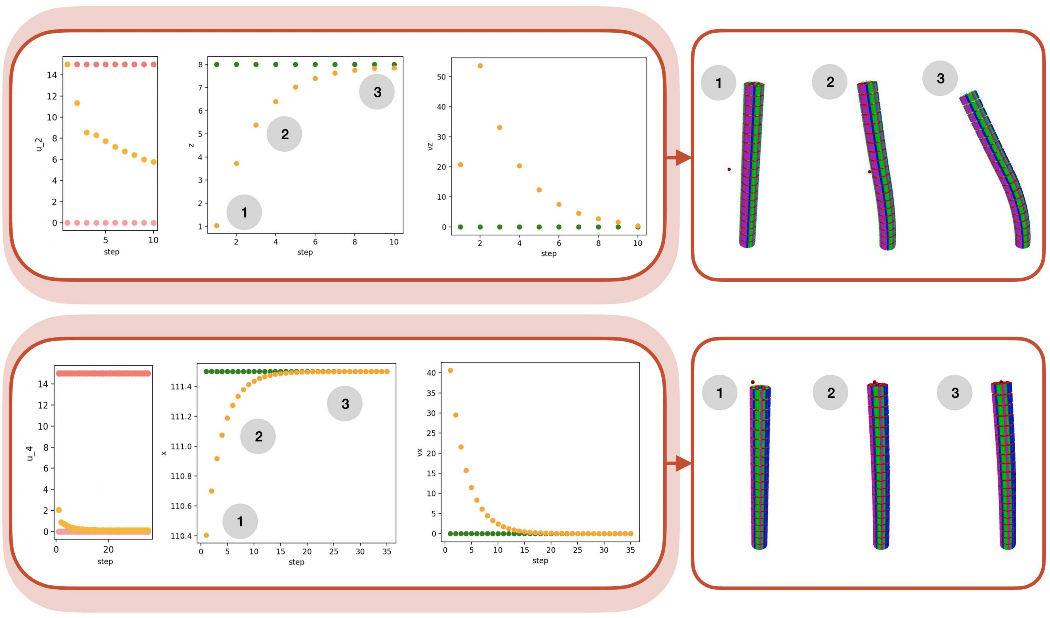


Fig. 5.21: Results obtained for the PneumaticTrunk. First row: Effector at the middle of the robot. Second row: Effector at the end of the robot. From left to right: pressure value (bar) for one of the actuators; position of the effectors along one axis (mm); velocity of the effectors along the same axis (mm/s); three configuration of the robots (1, 2, 3). Green: the objective. Orange: position and velocity of the effector. Pink: actuation bounds.

We can see that different deformations can be obtained, depending on the position of the end-effector and the target. In practice, all the positions in the robot's workspace can be reached if enough time steps are available to solve the task.

5.2.5 Scenario 3: Multiple goals with the Lianford robot.

The first two scenarios focus on positioning one effector with a maximum of two targets. In addition, the two previous robots are simulated with volumic FEM models, while the Liandford robot is simulated with a lattice of beam FEM elements. The third scenario involves optimizing a trajectory with multiple goals. We consider three different cases. The first is a spiral trajectory where each time step corresponds to a new goal. The second is a pick and place task where the end effector at the centre of the robot has to move back and forth between two positions. The last is an orientation task using two end-effectors at the same time. The results are shown in Figure 5.22.

For the first task, we can see that the robot can be controlled over several time steps, with a goal that varies at each time step. The obtained trajectory corresponds to the one that minimizes the cost function and follows the optimization constraints, at the tolerance level set. There are a few differences between the target trajectory and the computed trajectory: the target trajectory only takes into account

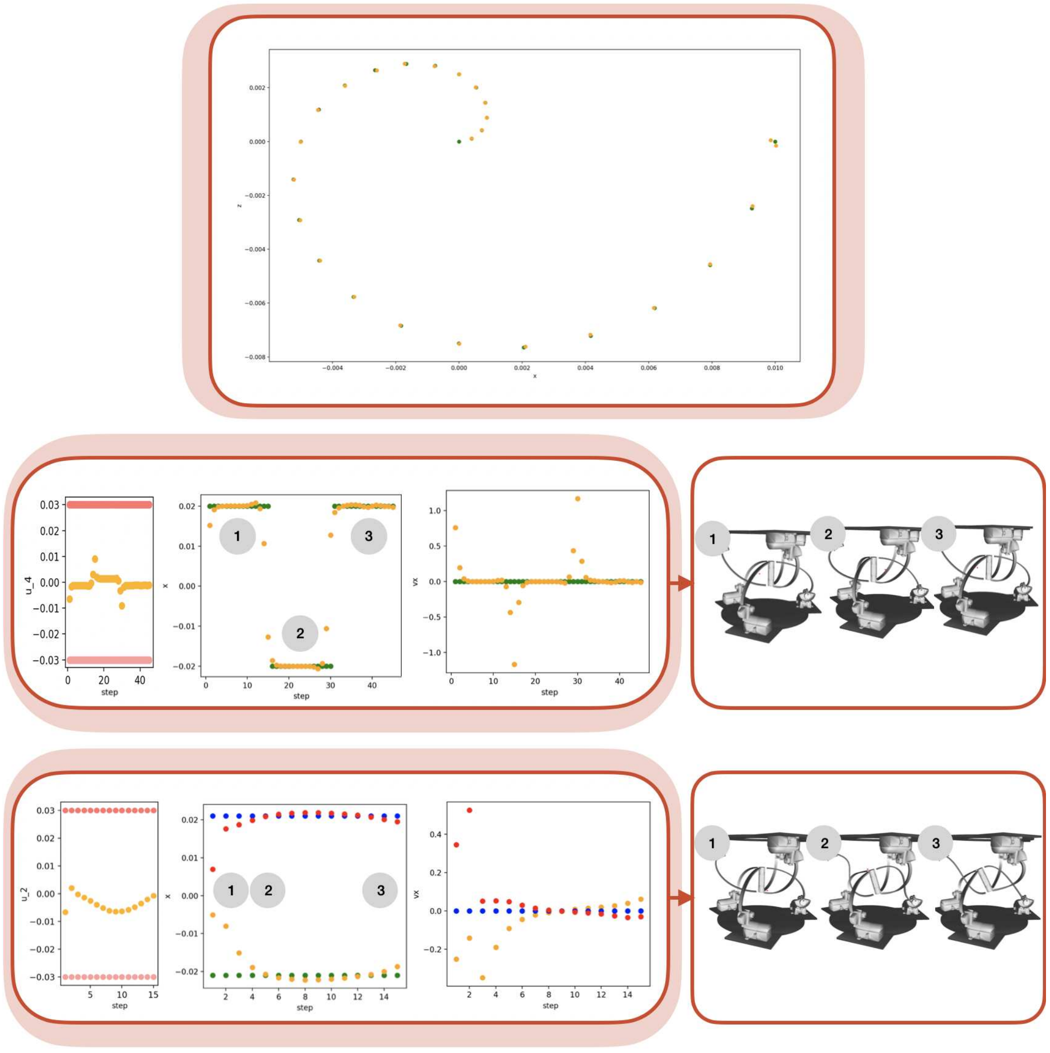


Fig. 5.22: Results obtained for the Lianford robot control. First row: Position on the x-z plane (mm) for the spiral trajectory. Green: the objective. Orange: position of the effector on the considering plane. Second row, from left to right: actuation (N), position (m), velocity (m/s), and three configuration of the robot for the pick and place task. Green: the objective. Orange: position and velocity of the effector. Pink: actuation bounds. Third row, from left to right: actuation (N), position (m), velocity (m/s), and three configuration of the robot for the pick and place task. Green and Blue: the objective. Orange and Red: position and velocity of the effector. Pink: actuation bounds.

the positioning of the end-effector, whereas the computed trajectory minimises an objective that includes not only the position, but also the velocity of the effector and the actuation value. Changing the weight of these elements in the loss leads to different results. Here, the emphasis is on positioning. The second task shows that it is possible to control the robot's dynamics on more complex trajectories, such as tasks where it has to navigate between two different positions. Finally, the third task shows that it is possible to control several effectors at the same time.

5.2.6 Discussion

We have proposed a method for optimizing the trajectory of soft robots using dynamic FEM models. This method is based on the joint use of a full FEM model to obtain a nominal trajectory in the forward part, and the use of a condensed FEM model to find the actuation in the backward part. This method can handle several types of robot geometry (parallel or slender), several type of FEM elements (tetra, hexa, beam), several types of actuation (cable, pneumatic, joint) and several types of task (one goal, several goals, one trajectory). We have used three different examples to illustrate all these features.

FEM simulation is known to be slower than other methods such as PWCC, but it can faithfully simulate robots with more complex geometry or actuation. It would be possible to use faster models in forward, such as model reduction. The advantage of this method over previous ones is that the optimization does not depend on the choice of model used for the simulation, the condensation being valid for FEM models, reduced models or beam models.

The fact that only the derivative of the linear part of the dynamics is used in the backward step means that no additional computations are required compared with the forward step, and speeds up the convergence of the method. The Diamond example showed that the two approaches lead to similar results. We found that the non-linear part tended to degrade convergence, which is surprising. We have not yet been able to explain this problem. However, the fact that the problem converges with just the linear part in the 3 cases we studied is very positive as it is more efficient in terms of calculation time. In addition, the dynamics used in the forward stage and the linearised condensed dynamics used in the backward stage are not the same. This leads to different gradients between those calculated from the position of the end-effectors and those calculated from the condensed model. This can lead to the framework's dual error not converging, although in practice this does not affect the control of the robots.

It is also possible to extend the capabilities of the method, in particular by adding different constraints. For example, it would be possible to imagine constraints on the maximum displacement increment of actuators, or constraints on the position of the end effector by defining zones where it is not allowed to go. Another extension would be to consider perturbations in the trajectory. This would enable the development of tasks where a force is applied to the robot at a given point in the trajectory. Finally, we are using a line search approach to update the variables, but other methods such as trust region approaches could be considered.

5.3 Conclusion

In this Chapter we have demonstrated that the condensed FEM model can be used to describe the state of the soft robot in the constraint space. This model is sufficiently expressive to control (both with and without contact) the soft robot, and when learned, its differentiability allows for calibration and design parameter optimisation. Additionally, the prediction is made with a small-scale network, making it well suited for embedded control algorithms on microprocessors, on which it would not be possible to compute full FEM models. The prediction is fast and control can be performed in real time. Finally, the condensed model can also be used to formulate trajectory optimisation problems using optimal control formalism, allowing for the control of the soft robot's position and velocity over several time steps.

However, this approach has some limitations. In the context of the learned model, a large amount of data is required for learning. The amount of data increases with the complexity of the task, such as control in a large workspace or multiparametric design optimisation. Moreover, the prediction performance of the learned model depends on sampling. To address this issue, it would be beneficial to explore the possibility of active sampling to update the learned model according to the robot configurations encountered during exploitation. In the context of trajectory optimisation, the forward phase involves computing the full robot dynamics, which can be slow when the underlying FEM model is complex. However, by focusing on the linear part of the problem in the backward phase, we could explore real-time applications in which the forward phase is performed with a learned condensed model.

6 Conclusion

Contents

7. Rippling.

— **Nemo**

And what remains at the end? After a 3-year PhD thesis concludes, after a romance terminates, or even after life itself? There is a captivating concept known as Rippling theory, which suggests that we persist through the effects we had while we were present. There is education we impart to our children, the lessons we teach as professors, the amiability we share with strangers on the street. There are collaborations with colleagues, the research we produce as scientists, and the art that we create. Just as a drop of water on a tranquil lake continues to have an impact after it has dispersed, what we share continues even after we have gone.

The research presented in this PhD thesis can be divided into three main categories: modelling, simulation, and control. In the first part of the conclusion (6.1), the contributions of the manuscript are summarized according to these categories. The relevance and limitations of the methods developed in this study are discussed. Finally, the prospects of this research in both the short- and long-term are examined in Section (6.2).

6.1 Conclusion on the contributions

6.1.1 Modelling

In this PhD thesis, we modeled soft robots for control and design tasks. In particular, we explored different methods for describing the state of soft robots. In Chapter 3, the state of the robot is described using the FEM model by selecting key points on the robot. In Chapter 4, two strategies for simplifying the description of a robot's state are developed. The first method involves using a proxy model to describe a soft robot in its configuration space. The second method involves partitioning the robot's observation space into different contact configuration spaces. Finally, in Chapter 5, the state of the robot is described using a condensed FEM model in the constraint space.

These models allow to explore the links between soft robotics and artificial intelligence. Full modelling of the soft robot is not useful at all stages of learning. With the proxy, we can see that modelling and simulating the deformations of the robot's skeleton are enough to generate trajectories for control purposes. The graph of configuration spaces demonstrates that it is not necessary to learn the behaviour of the entire system but rather how to move from one contact configuration to another. It is possible to model and learn how to solve the Maze before using other tools to solve it with a Tripod robot. Finally, only the condensed model is used in single- or multi-step control schemes. Through these various examples, we demonstrate that it is possible to be much more subtle than the opposition between physical modelling and learning. There are many ways of combining the different models in order to increase efficiency (in terms of simulation time and resources, as well as sometimes in terms of learning quality).

6.1.2 Simulation

This research on various soft robot models leads to the production of a significant amount of open-source code. Notable achievements include the development of SofaGym, the creation of a condensed FEM model learning pipeline, and the implementation of a plugin that enables trajectory optimisation from a soft robot FEM model.

First, all the results obtained with RL and planning algorithms are based on SofaGym. SofaGym is the first software to integrate RL, planning and FEM simulations of soft robots. It was developed when few software solutions were available for these challenges, or when existing solutions had limited capabilities in simulating complex robots. SofaGym is built on Gym, a reference API that facilitates the use of classical RL algorithms. This software is employed in Chapter 3 to demonstrate the possibility of linking RL and FEM simulations. It is also used in Chapter 4 to generate a trajectory with a proxy and to learn how to solve tasks using a graph of configuration spaces.

Chapter 5 focuses on the use of a condensed FEM model for control and design applications. Learning this condensed model introduces neural networks into the direct and inverse simulations of soft robots. These neural networks can rapidly predict the mechanical matrices used in control and design problems, enabling applications such as embedded control or optimisation of design parameters. The simulation pipeline developed for this method is distinct from a conventional FEM simulation. Data are first collected and the networks are trained using these data. Then, the FEM simulation can be entirely bypassed, and the networks can be used to simulate the mechanical quantities of soft robots. The condensed model is also used for trajectory optimisation. The software developed for trajectory simulation can simulate soft robots from any position on the nominal trajectory and retrieve the mechanical quantities required for optimisation. This is the first software package that enables trajectory optimisation from a FEM model of a soft robot using the DDP method.

6.1.3 Control

The three chapters, Chapter 3, Chapter 4, and Chapter 5, explore the relationship between various methods for describing a robot's state and its control. The general idea is to control soft robots to perform high level tasks. In Chapter 3, the study of SofaGym demonstrates that FEM models of soft robots can be controlled using either RL algorithms or optimisation methods combined with inverse robot modelling. However, this chapter also highlights the limitations of these methods. RL algorithms require a significant amount of training data, which are expensive to obtain with FEM models, and are not always interpretable. Optimisation-based methods cannot solve high-level tasks alone and may sometimes result in local minima.

Based on the examples of CartStemContact, Multigait and Maze, we can identify what is interesting to learn and what is interesting to calculate using optimisation methods. In Chapter 4, the proxy approach separates the control problem into two stages: generating an admissible trajectory using a simplified model of the robot, and tracking and correcting this trajectory using the FEM model of the robot. The graph approach uses optimisation methods when possible and learns action sequences to change the contact configuration when not. These two approaches effectively combine learning and optimisation approaches and address the challenges presented at the begining of this work.

However, these approaches rely on simplifying the learning stage, either by using a simplified model to represent the robot or by incorporating prior knowledge into the learning process using the configuration spaces graph approach. To overcome this problem, we proposed in Chapter 5 the use of a condensed FEM model of the robot. This is a low-dimensional model that learns mechanical quantities that synthesize the link between constraints in the robot rather than directly learning the inverse model of the robot. It can be learned for control tasks without contact, with contact, and even as a substitute for the FEM simulation. It can also be used in a multi-step control approach, as proposed in the optimal control framework.

6.1.4 Relevance and limitations

This study demonstrates the strengths and weaknesses of various control approaches proposed for soft robotics. By focusing on modelling and describing the state of a soft robot, it is possible to effectively control soft robots to perform high level task. Some descriptions also have broader applications, such as the learned condensed FEM model, which can be used for calibration and parameter optimisation. This approach of "learning what is necessary and leaving the rest" has guided this PhD thesis and resulted in the development of new, systematic methods for control issues in soft robotics.

However, this study has some limitations. First, there are no significant testing on physical prototypes. Although FEM models have been used in the past to control physical prototypes, and all the work is based on these models, only the passage from simulation to reality can fully validate the tools presented in this work. However, the passage to reality presents its own challenges that need to be considered, such as the detection of contact configurations from sensor data. These issues offer a wide range of research possibilities outside the scope of this study. Finally, in this work we have defined high-level tasks as generating trajectories for positioning, manipulation or locomotion. However, we are far from the high-level tasks proposed in the rigid robotics literature, like cooking for example. We propose methods to take a further step in this direction, and further studies are needed to reach this level of capability for soft robots.

6.2 Perspectives and futur works

6.2.1 Short-term perspective: implicit parameterization and MPC

The work proposed in this study offers numerous development prospects. First, two use cases of the condensed FEM model are presented in Chapter 5. In the framework of trajectory optimisation, we showed that it is possible to only use derivatives of the linear part of the dynamics in the backward step. These derivatives involve only W the admittance matrix projected into constraint space. This is possible because the forward step captures the nonlinearities of the dynamics in order to build the nominal trajectory. The learned condensed FEM model can be integrated into both the forward and backward calculations. Nonlinearities in the dynamics are considered in the forward step using δ_e^{free}. In addition, given the displacement of the actuators, it is possible to calculate the W matrice required for backward computation of the framework. Integrating this learned condensed FEM model into the trajectory optimisation framework makes it possible to imagine MPC-type real-time control applications for physical prototypes.

The second element that can be explored is related to the design of the soft robots. In this study, we investigated two design applications using the condensed FEM model (calibration and parameter optimisation). This work uses a parametric model of the robot [Nav+23]. However, the choice of these parameters limits the available shape of the robot. Generative models [RNA22; Bon+22] can be used to generate realistic soft robot shapes from an existing robot shapes dataset. Using local rules such as Neural Cellular Automata [Mor+20; Sud+21], it is possible to imagine a framework in which realistic robot patches are generated using low-dimensional networks before being combined and assembled using evolutionary algorithms and topological optimisation schemes. The robot's state is described implicitly, as a collection of realistic patches. This would generate robots that look like the robots we know but are optimized for a given geometric or mechanical task.

6.2.2 Medium- and long-term perspective: differentiable simulation, perception and self-representation

From a slightly longer-term perspective, other elements can be explored. For instance, work on the simulation differentiability could improve the design and control applications. The design applications discussed in Chapter 5 are enabled by a differentiable network that connects design parameters to mechanical quantities. Research on simulators could result in the ability to obtain gradient-based parameter optimisation methods without first having to learn a condensed model. This topic has been explored using different types of simulators, such as the MPM simulator presented in Chapter 2 [Mat+23]. From a control perspective, differentiability enables the integration of gradient information into the control methods. Specifically, contact differentiability, as proposed for rigid robotics [Mon+22], would allow for the integration of 1st-order contact information, which would change the way we understand the contact reconfiguration.

Although we have not explored the topic of perception in this PhD study, it is an important aspect that is fully integrated in describing the state of robots. Specifically, it answers the question of how the robot describes itself using the data given by the sensors. An interesting idea is to draw inspiration from both morphological computation [Hau+11] and the holographic principle defined for biological systems [Fie+23]. The idea is to represent the robot's mind as a deformable structure that changes according to the information coming from the sensors, leading to describe the state of the sensors in a condensed manner. This idea can be applied to any type of robotic system, whether rigid, soft, or biological [EL21; Bla+22].

Bibliography

[AD19] Sanjeevan Ahilan and Peter Dayan. „Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning“. In: *ArXiv* abs/1901.08492 (2019) (cit. on p. 25).

[Agr+16] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. „Learning to Poke by Poking: Experiential Learning of Intuitive Physics“. In: *CoRR* abs/1606.07419 (2016). arXiv: 1606.07419 (cit. on p. 20).

[Akr+18] Riad Akrour, Filipe Veiga, Jan Peters, and Gerhard Neumann. „Regularizing Reinforcement Learning with State Abstraction“. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 534–539 (cit. on p. 25).

[Alm+21] João Damião Almeida, Paul Schydlo, Atabak Dehban, and José Santos-Victor. „Sensorimotor Graph: Action-Conditioned Graph Neural Network for Learning Robotic Soft Hand Dynamics“. In: *CoRR* abs/2107.08492 (2021). arXiv: 2107.08492 (cit. on p. 36).

[Alo+22] John Irvin Alora, Mattia Cenedese, Edward Schmerling, George Haller, and Marco Pavone. „Data-Driven Spectral Submanifold Reduction for Nonlinear Optimal Control of High-Dimensional Robots“. In: 2022, pp. 2627–2633 (cit. on p. 31).

[And+17] Marcin Andrychowicz, Dwight Crow, Alex Ray, et al. „Hindsight Experience Replay“. In: vol. abs/1707.01495. 2017 (cit. on p. 84).

[Ant73] Stuart S. Antman. „The Theory of Rods“. In: 1973 (cit. on p. 15).

[AZK21] Walid Amehri, Gang Zheng, and Alexandre Kruszewski. „Workspace Boundary Estimation for Soft Manipulators Using a Continuation Approach“. In: vol. 6. 2021, pp. 7169–7176 (cit. on p. 66).

[AZK22] Walid Amehri, Gang Zheng, and Alexandre Kruszewski. „FEM-Based Exterior Workspace Boundary Estimation for Soft Robots via Optimization“. In: vol. 7. 2022, pp. 3672–3678 (cit. on p. 66).

[Bam+22] Antoine Bambade, Sarah El-Kazdadi, Adrien Taylor, and Justin Carpentier. „PROX-QP: Yet another Quadratic Programming Solver for Robotics and beyond“. In: 2022 (cit. on p. 100).

[Bel57] Richard Bellman. „A Markovian Decision Process“. In: vol. 6. 5. Indiana University Mathematics Department, 1957, pp. 679–684 (cit. on p. 24).

[Ber+15] Christos Bergeles, Andrew H. C. Gosline, Nikolay V. Vasilyev, et al. „Concentric Tube Robot Design and Optimization Based on Task and Anatomical Constraints“. In: vol. 31. 2015, pp. 67–84 (cit. on p. 34).

[Ber+20] James M. Bern, Yannick Schnider, P. Banzet, Nitish Kumar, and Stelian Coros. „Soft Robot Control With a Learned Differentiable Model“. In: 2020, pp. 417–423 (cit. on p. 36).

[Bha+19a] Sarthak Bhagat, Hritwick Banerjee, Zion Ho Tse, and Hongliang Ren. „Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges“. en. In: *Robotics* 8.1 (Jan. 2019), p. 4 (cit. on p. 20).

[Bha+19b] Sarthak Bhagat, Hritwick Banerjee, Zion Tsz Ho Tse, and Hongliang Ren. „Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges“. In: *Robotics* 8 (2019), p. 4 (cit. on p. 25).

[Bla+22] D. Blackiston, S. Kriegman, J. Bongard, and M. Levin. „Biological Robots: Perspectives on an Emerging Interdisciplinary Field“. In: 2022. arXiv: 2207.00880 [cs.RO] (cit. on p. 141).

[BM10] Sébastien Bubeck and Rémi Munos. „Open Loop Optimistic Planning“. In: *Annual Conference Computational Learning Theory*. 2010 (cit. on p. 27).

[Bon+22] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. „Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models“. In: vol. 44. 11. Institute of Electrical and Electronics Engineers (IEEE), Nov. 2022, pp. 7327–7347 (cit. on p. 140).

[Bou+18] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, et al. „Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping“. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 4243–4250 (cit. on p. 25).

[Bra+07] David Braganza, Darren M. Dawson, Ian D. Walker, and Nitendra Nath. „A Neural Network Controller for Continuum Robots“. In: *IEEE Transactions on Robotics* 23 (2007), pp. 1270–1277 (cit. on p. 20).

[Bro+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, et al. „Openai gym“. In: 2016 (cit. on p. 16).

[Bud+18] Rohan Budhiraja, Justin Carpentier, Carlos Mastalli, and Nicolas Mansard. „Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics“. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)* (2018), pp. 1–9 (cit. on pp. 28, 36, 37).

[CB21] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2021 (cit. on p. 16).

[CED17] Eulalie Coevoet, Adrien Escande, and Christian Duriez. „Optimization-Based Inverse Model of Soft Robots With Contact Handling“. In: *IEEE Robotics and Automation Letters* PP (Feb. 2017), pp. 1–1 (cit. on pp. 4, 8, 31, 46).

[CED19] Eulalie Coevoet, Adrien Escande, and Christian Duriez. „Soft robots locomotion and manipulation control using FEM simulation and quadratic programming". In: *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE. 2019, pp. 739–745 (cit. on pp. 15, 20, 21).

[Cha+23] Paul Chaillou, Jialei Shi, Alexandre Kruszewski, et al. „Reduced finite element modelling and closed-loop control of pneumatic-driven soft continuum robots". In: *2023 IEEE International Conference on Soft Robotics (RoboSoft)*. 2023, pp. 1–8 (cit. on p. 111).

[Che+18] Feifei Chen, Wenjun Xu, Hongying Zhang, and al. „Topology Optimized Design, Fabrication, and Characterization of a Soft Cable-Driven Gripper". In: *IEEE Robotics and Automation Letters* 3 (2018), pp. 2463–2470 (cit. on p. 34).

[Che+21] ShiTong Chen, YuSheng Wang, DeChen Li, FeiFei Chen, and XiangYang Zhu. „Enhancing interaction performance of soft pneumatic-networks grippers by skeleton topology optimization". In: *Science China Technological Sciences* 64.12 (2021), pp. 2709–2717 (cit. on p. 34).

[Cho+20] Heesun Choi, Cindy Crump, Christian Duriez, et al. „On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward". In: *Proceedings of the National Academy of Sciences of the United States of America* 118 (2020) (cit. on p. 15).

[Cho+21] HeeSun Choi, Cindy Crump, Christian Duriez, et al. „On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward". In: *Proceedings of the National Academy of Sciences* 118.1 (2021). eprint: `https://www.pnas.org/content/118/1/e1907856118.full.pdf` (cit. on p. 21).

[Cia+14] Matteo Cianchetti, Tommaso Ranzani, Giada Gerboni, et al. „Soft Robotics Technologies to Address Shortcomings in Today ' s Minimally Invasive Surgery : The STIFF-FLOP Approach". In: 2014 (cit. on p. 111).

[Cia+18] Matteo Cianchetti, Cecilia Laschi, Arianna Menciassi, and Paolo Dario. „Biomedical applications of soft robotics". In: vol. 3. 2018, pp. 143–153 (cit. on p. 3).

[Coe+17] Eulalie Coevoet, Thor Morales-Bieze, Frederick Largilliere, et al. „Software toolkit for modeling, simulation, and control of soft robots". In: *Advanced Robotics* 31.22 (2017), pp. 1208–1224 (cit. on pp. 14, 17).

[Coe19] Eulalie Coevoet. „Optimization-based inverse model of soft robots, with contact handling". In: 2019 (cit. on pp. 3, 20, 31).

[CW20] Feifei Chen and Michael Yu Wang. „Design Optimization of Soft Robots: A Review of the State of the Art". In: *IEEE Robotics Automation Magazine* 27.4 (2020), pp. 27–43 (cit. on p. 33).

[DH92] Peter Dayan and Geoffrey E. Hinton. „Feudal Reinforcement Learning". In: *NIPS*. 1992 (cit. on p. 25).

[DI19] Zhuoran Dang and Mamoru Ishii. „A physics-informed reinforcement learning approach for the interfacial area transport in two-phase flow". In: *arXiv: Computational Physics* (2019) (cit. on p. 25).

[Dij59] Edsger W. Dijkstra. „A note on two problems in connexion with graphs". In: vol. 1. 1959, pp. 269–271 (cit. on p. 48).

[DJA22] Fei Deng, Ingook Jang, and Sungjin Ahn. *DreamerPro: Reconstruction-Free Model-Based Reinforcement Learning with Prototypical Representations*. 2022 (cit. on p. 25).

[Dur+18] Christian Duriez, Hadrien Courtecuisse, Juan-Pablo de La Plata Alcalde, and Pierre-Jean Bensoussan. „Contact Skinning“. In: 2018 (cit. on p. 68).

[Dur13] Christian Duriez. „Control of elastic soft robots based on real-time finite element method“. In: *2013 IEEE International Conference on Robotics and Automation* (2013), pp. 3982–3987 (cit. on pp. 15, 46).

[DVT14] Hongkai Dai, Andres K. Valenzuela, and Russ Tedrake. „Whole-body motion planning with centroidal dynamics and full kinematics“. In: 2014, pp. 295–302 (cit. on p. 36).

[Ebe+21] Imme Ebert-Uphoff, Ryan Lagerquist, Kyle Hilburn, et al. „CIRA Guide to Custom Loss Functions for Neural Networks in Environmental Sciences - Version 1“. In: *CoRR* abs/2106.09757 (2021). arXiv: `2106.09757` (cit. on p. 24).

[EL21] Mohammed Ebrahimkhani and Michael Levin. „Synthetic living machines: A new window on life“. In: vol. 24. 2021 (cit. on p. 141).

[ELS21] Benjamin Eysenbach, Sergey Levine, and Ruslan Salakhutdinov. „Replacing Rewards with Examples: Example-Based Policy Search via Recursive Classification“. In: *NeurIPS*. 2021 (cit. on p. 25).

[Fa12] Hervé Delingette François Faure Christian Duriez and al. „SOFA: A Multi-Model Framework for Interactive Physical Simulation“. In: *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery* 11 (2012), pp. 283–321 (cit. on pp. 17, 20).

[Fal+15] Valentin Falkenhahn, Alexander Hildebrandt, Rudiger Neumann, and Oliver Sawodny. „Model-based feedforward position control of constant curvature continuum robots using feedback linearization“. en. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA: IEEE, May 2015, pp. 762–767 (cit. on p. 16).

[FDA17] Carlos Florensa, Yan Duan, and P. Abbeel. „Stochastic Neural Networks for Hierarchical Reinforcement Learning“. In: *ArXiv* abs/1704.03012 (2017) (cit. on p. 25).

[Fie+23] Chris Fields, Filippo Fabrocini, Karl Friston, et al. „Control flow in active inference systems“. In: 2023. arXiv: `2303.01514 [q-bio.NC]` (cit. on p. 141).

[FL16] Chelsea Finn and Sergey Levine. „Deep visual foresight for planning robot motion“. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2016), pp. 2786–2793 (cit. on p. 24).

[Fox+17] Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. „Multi-Level Discovery of Deep Options“. In: *ArXiv* abs/1703.08294 (2017) (cit. on p. 25).

[GC14] Tipakorn Greigarn and M. Cenk Cavusoglu. „Task-space motion planning of MRI-actuated catheters for catheter ablation of atrial fibrillation“. en. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Chicago, IL, USA: IEEE, Sept. 2014, pp. 3476–3482 (cit. on p. 16).

[GD18] Olivier Goury and Christian Duriez. „Fast, generic and reliable control and simulation of soft robots using model order reduction“. In: *IEEE Transactions on Robotics* (2018) (cit. on pp. 8, 36, 43, 49).

[GD19] S. Gros and M. Diehl. „Numerical optimal control“. In: 2019 (cit. on p. 28).

[Geo+16] Thomas George Thuruthel, Egidio Falotico, Matteo Cianchetti, and Cecilia Laschi. „Learning Global Inverse Kinematics Solutions for a Continuum Robot“. In: vol. 569. Jan. 2016, pp. 47–54 (cit. on p. 15).

[GJC09] Yusof Ganji, Farrokh Janabi-Sharifi, and Asim N. Cheema. „Robot-assisted catheter manipulation for intracardiac navigation“. en. In: *International Journal of Computer Assisted Radiology and Surgery* 4.4 (June 2009), pp. 307–315 (cit. on p. 16).

[Gra+21] Moritz A. Graule, Clark B. Teeple, Thomas P. McCarthy, et al. „SoMo: Fast and Accurate Simulations of Continuum Robots in Complex Environments“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*. IEEE, 2021, pp. 3934–3941 (cit. on pp. 16, 17).

[Gu+19] Yueqin Gu, Xuecheng Zhang, Qiuxuan Wu, et al. „Research on Motion Evolution of Soft Robot Based on VoxCAD“. In: *International Conference on Intelligent Robotics and Applications*. 2019 (cit. on p. 34).

[Gul90] Vijaykumar Gullapalli. „A stochastic reinforcement learning algorithm for learning real-valued functions“. In: *Neural networks* 3.6 (1990), pp. 671–692 (cit. on p. 24).

[Gup+19] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. „Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning“. In: *CoRL*. 2019 (cit. on p. 25).

[Haa+18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. „Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor“. In: *CoRR* abs/1801.01290 (2018). arXiv: `1801.01290` (cit. on p. 24).

[Haf+18] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, et al. „Learning Latent Dynamics for Planning from Pixels“. In: *CoRR* abs/1811.04551 (2018). arXiv: `1811.04551` (cit. on p. 20).

[Hau+11] Helmut Hauser, Auke Jan Ijspeert, Rudolf Marcel Füchslin, Rolf Pfeifer, and Wolfgang Maass. „Towards a theoretical foundation for morphological computation with compliant bodies“. In: vol. 105. 2011, pp. 355–370 (cit. on p. 141).

[Haw+19] Zachary Hawks, Chase G. Frazelle, Keith Evan Green, and Ian D. Walker. „Motion Planning for a Continuum Robotic Mobile Lamp: Defining and Navigating the Configuration Space“. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), pp. 2559–2566 (cit. on p. 20).

[Hen17] Bernhard Hengst. „Hierarchical Reinforcement Learning“. In: Jan. 2017 (cit. on p. 25).

[Hes+13] Hesheng Wang, Weidong Chen, Xiaojin Yu, et al. „Visual servo control of cable-driven soft robotic manipulator“. en. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo: IEEE, Nov. 2013, pp. 57–62 (cit. on p. 16).

[HGS15] H. V. Hasselt, Arthur Guez, and David Silver. „Deep Reinforcement Learning with Double Q-Learning“. In: *ArXiv* abs/1509.06461 (2015) (cit. on p. 24).

[HM08] Jean-François Hren and Rémi Munos. „Optimistic Planning of Deterministic Systems“. In: *European Workshop on Reinforcement Learning*. 2008 (cit. on p. 27).

[Hom+18] Bianca Homberg, Robert K. Katzschmann, Mehmet Remzi Dogar, and Daniela Rus. „Robust proprioceptive grasping with a soft robot hand“. In: *Autonomous Robots* 43 (2018), pp. 681–696 (cit. on p. 20).

[How60] R. A. Howard. „Dynamic Programming and Markov Processes“. In: Cambridge, MA: MIT Press, 1960 (cit. on p. 24).

[Hu+19a] Yeping Hu, Alireza Nakhaei, Masayoshi Tomizuka, and Kikuo Fujimura. „Interaction-aware Decision Making with Adaptive Strategies under Merging Scenarios“. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), pp. 151–158 (cit. on p. 25).

[Hu+19b] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, et al. „ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics“. In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), pp. 6265–6271 (cit. on p. 17).

[Hu+20] Yuanming Hu, Luke Anderson, Tzu-Mao Li, et al. „DiffTaichi: Differentiable Programming for Physical Simulation“. In: *ArXiv* abs/1910.00935 (2020) (cit. on p. 17).

[Hua+21a] Jiang Hua, Liangcai Zeng, Gongfa Li, and Zhaojie Ju. „Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning“. In: *Sensors (Basel, Switzerland)* 21 (2021) (cit. on p. 25).

[Hua+21b] Zhiao Huang, Yuanming Hu, Tao Du, et al. *PlasticineLab: A Soft-Body Manipulation Benchmark with Differentiable Physics*. Apr. 2021 (cit. on p. 16).

[Jal+22] Wilson Jallet, Antoine Bambade, Nicolas Mansard, and Justin Carpentier. „Constrained Differential Dynamic Programming: A primal-dual augmented Lagrangian approach“. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Kyoto, Japan, Oct. 2022 (cit. on p. 128).

[JH21] Anthony Vanderkop Jack Collins Shelvin Chand and David Howard. „A Review of Physics Simulators for Robotic Applications“. In: *IEEE Access* 9 (2021), pp. 51416–51431 (cit. on p. 16).

[Jia+21] Xiaowei Jia, Jared D. Willard, Anuj Karpatne, et al. „Physics-Guided Machine Learning for Scientific Discovery: An Application in Simulating Lake Temperature Profiles“. In: *ACM/IMS Transactions on Data Science* 2 (2021), pp. 1–26 (cit. on p. 25).

[JMC22] Wilson Jallet, Nicolas Mansard, and Justin Carpentier. „Implicit Differential Dynamic Programming“. In: *International Conference on Robotics and Automation (ICRA 2022)*. IEEE Robotics and Automation Society. Philadelphia, United States, May 2022 (cit. on pp. 28–30).

[JP13] J. Andrew Bagnell Jens Kober and Jan Peters. „Reinforcement learning in robotics: A survey“. In: *International Journal of Robotics Research* 32 (2013), pp. 1238–1274 (cit. on p. 20).

[Kah+20] Adar Kahana, Eli Turkel, Shai Dekel, and Dan Givoli. „Obstacle segmentation based on the wave equation and deep learning“. In: *J. Comput. Phys.* 413 (2020), p. 109458 (cit. on p. 24).

[Kar+17] Anuj Karpatne, Gowtham Atluri, James H. Faghmous, et al. „Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data“. In: *IEEE Transactions on Knowledge and Data Engineering* 29 (2017), pp. 2318–2331 (cit. on p. 24).

[KG17] Wah Loon Keng and Laura Graesser. „SLM Lab“. In: GitHub, 2017 (cit. on p. 84).

[Kim+21a] Daekyum Kim, Sang-Hun Kim, Taekyoung Kim, et al. „Review of machine learning methods in soft robotics“. In: *PLoS ONE* 16 (2021) (cit. on pp. 14, 15).

[Kim+21b] J. Taery Kim, Jeongeun Park, Sungjoon Choi, and Sehoon Ha. „Learning Robot Structure and Motion Embeddings using Graph Neural Networks“. In: *ArXiv* abs/2109.07543 (2021) (cit. on p. 36).

[KP13a] Mahta Khoshnam and Rajni V. Patel. „A pseudo-rigid-body 3R model for a steerable ablation catheter“. en. In: *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany: IEEE, May 2013, pp. 4427–4432 (cit. on p. 16).

[KP13b] Mahta Khoshnam and Rajnikant V. Patel. „A pseudo-rigid-body 3R model for a steerable ablation catheter“. In: *2013 IEEE International Conference on Robotics and Automation* (2013), pp. 4427–4432 (cit. on p. 15).

[KS06] Levente Kocsis and Csaba Szepesvári. „Bandit Based Monte-Carlo Planning“. In: *Proc. European Conference on Machine Learning (ECML)*. Vol. 2006. Sept. 2006, pp. 282–293 (cit. on p. 27).

[Kul+16] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. „Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation“. In: *NIPS*. 2016 (cit. on p. 25).

[LaV06] Steven LaVallée. *Planning Algorithms*. available freely online at `http://lavalle.pl/planning/`. Cambridge University Press, 2006 (cit. on p. 27).

[LCY22] Chih-Hsing Liu, Yang Chen, and Sy-Yeu Yang. „Topology Optimization and Prototype of a Multimaterial-Like Compliant Finger by Varying the Infill Density in 3D Printing“. In: *Soft Robotics* 9.5 (Oct. 2022), pp. 837–849 (cit. on p. 34).

[Les+17] Timothée Lesort, Mathieu Seurin, Xinrui Li, Natalia Díaz Rodríguez, and David Filliat. „Unsupervised state representation learning with robotic priors: a robustness benchmark“. In: *ArXiv* abs/1709.05185 (2017) (cit. on p. 36).

[Les+18] Timothée Lesort, Natalia Díaz Rodríguez, Jean-François Goudou, and David Filliat. „State Representation Learning for Control: An Overview“. In: *Neural networks : the official journal of the International Neural Network Society* 108 (2018), pp. 379–392 (cit. on pp. 35, 36).

[Leu20] Edouard Leurent. „Safe and Efficient Reinforcement Learning for Behavioural Planning in Autonomous Driving“. In: 2020 (cit. on p. 27).

[Lev+16] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. „Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection“. en. In: *arXiv:1603.02199 [cs]* (Aug. 2016). arXiv: 1603.02199 (cit. on p. 20).

[Lev+19] Andrew Levy, George Dimitri Konidaris, Robert W. Platt, and Kate Saenko. „Learning Multi-Level Hierarchies with Hindsight“. In: *ICLR*. 2019 (cit. on p. 25).

[Lid+22] Quentin Le Lidec, Louis-Rene Montaut, Cordelia Schmid, Ivan Laptev, and Justin Carpentier. „Leveraging Randomized Smoothing for Optimal Control of Nonsmooth Dynamical Systems“. In: vol. abs/2203.03986. 2022 (cit. on p. 80).

[Lil+15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, et al. „Continuous control with deep reinforcement learning“. In: *CoRR* abs/1509.02971 (2015) (cit. on p. 24).

[Lin+20] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. „SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation“. In: *CoRL*. 2020 (cit. on p. 16).

[Lin92] Long-Ji Lin. „Reinforcement learning for robots using neural networks“. PhD thesis. Carnegie Mellon University, 1992 (cit. on p. 24).

[Lip14] Hod Lipson. „Challenges and Opportunities for Design, Simulation, and Fabrication of Soft Robots“. In: 2014 (cit. on p. 14).

[Lof+15] Robert Tyler Loftin, Bei Peng, James MacGlashan, et al. „Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning“. In: *Autonomous Agents and Multi-Agent Systems* 30 (2015), pp. 30–59 (cit. on p. 25).

[Low+17] Ryan Lowe, Yi Wu, Aviv Tamar, et al. „Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments“. In: *ArXiv* abs/1706.02275 (2017) (cit. on p. 25).

[LS11] Kfir Yehuda Levy and Nahum Shimkin. „Unified Inter and Intra Options Learning Using Policy Gradient Methods“. In: *EWRL*. 2011 (cit. on p. 25).

[LSH07] Bryn A. Lloyd, Gábor Székely, and Matthias Harders. „Identification of Spring Parameters for Deformable Object Simulation“. In: *IEEE Transactions on Visualization and Computer Graphics* 13 (2007) (cit. on p. 16).

[LSH19] Brett T. Lopez, Jean-Jacques E. Slotine, and Jonathan P. How. „Dynamic Tube MPC for Nonlinear Systems“. In: 2019. arXiv: `1907.06553 [eess.SY]` (cit. on p. 80).

[Maj13] Carmel Majidi. „Soft Robotics : A Perspective — Current Trends and Prospects for the Future“. In: 2013 (cit. on p. 14).

[Man+04] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. „Dynamic abstraction in reinforcement learning via clustering“. In: *Proceedings of the twenty-first international conference on Machine learning* (2004) (cit. on p. 25).

[Mar+14] Andrew D. Marchese, Konrad Komorowski, Cagdas D. Onal, and Daniela Rus. „Design and control of a soft and continuously deformable 2D robotic manipulation system“. en. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, May 2014, pp. 2189–2196 (cit. on p. 16).

[Mar+22] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. „Motion Planning around Obstacles with Convex Optimization“. In: 2022. arXiv: `2205.04422 [cs.RO]` (cit. on p. 80).

[Mar+23] Tobia Marcucci, Jack Umenberger, Pablo A. Parrilo, and Russ Tedrake. „Shortest Paths in Graphs of Convex Sets“. In: 2023. arXiv: `2101.11565 [cs.DM]` (cit. on p. 80).

[Mas+17] Carlos Mastalli, Michele Focchi, Ioannis Havoutis, et al. „Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion“. In: 2017, pp. 1096–1103 (cit. on p. 36).

[Mas04] Michael Mascagni. „On the Scrambled Halton Sequence“. In: vol. 10. Dec. 2004, pp. 435–442 (cit. on p. 99).

[Mat+23] David Matthews, Andrew Spielberg, Daniela Rus, Sam Kriegman, and Josh Bongard. „Efficient automatic design of robots“. In: 2023. arXiv: `2306.03263 [cs.RO]` (cit. on p. 141).

[MAY73] DAVID Q. MAYNE. „Differential Dynamic Programming–A Unified Approach to the Optimization of Dynamic Systems* *This work was done during the author's visit to the Division of Engineering and Applied Physics, Harvard University, and was supported by the U.S. Army Research Office, the U.S. Air Force Office of Scientific Rearch and the U.S. Office of Naval Research under the Joint Services Electronics Program by Contracts N00014-67-A-0298-0006, 0005. and 0008.“ In: ed. by C.T. LEONDES. Vol. 10. Control and Dynamic Systems. Academic Press, 1973, pp. 179–254 (cit. on p. 29).

[MC16] Barbara Mazzolai and Matteo Cianchetti. „Soft robotics: Technologies and systems pushing the boundaries of robot abilities“. In: *Science Robotics* 1 (2016) (cit. on pp. 3, 67).

[MGK22] Brandon H. Meng, Isuru S. Godage, and Iyad Kang. „RRT*-Based Path Planning for Continuum Arms“. In: *IEEE Robot Autom Lett.* 7 (2022), pp. 6830–6937 (cit. on pp. 20, 68).

[MJD18] Jan Matas, Stephen James, and Andrew J. Davison. „Sim-to-Real Reinforcement Learning for Deformable Object Manipulation“. In: *CoRR* abs/1806.07851 (2018). arXiv: `1806.07851` (cit. on p. 16).

[MLH10] Igor Mordatch, Martin de Lasa, and Aaron Hertzmann. „Robust physics-based locomotion using low-dimensional planning“. In: 2010 (cit. on p. 36).

[MMD19] Thomas Morzadec, Damien Marchal, and Christian Duriez. „Toward Shape Optimization of Soft Robots“. In: *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)* (2019), pp. 521–526 (cit. on p. 34).

[Mni+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. „Playing Atari with Deep Reinforcement Learning“. In: *ArXiv* abs/1312.5602 (2013) (cit. on p. 24).

[Mni+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. „Human-level control through deep reinforcement learning“. In: *nature* 518.7540 (2015), pp. 529–533 (cit. on p. 20).

[Mni+16] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, et al. „Asynchronous Methods for Deep Reinforcement Learning“. In: *ArXiv* abs/1602.01783 (2016) (cit. on p. 24).

[Mon+22] Louis Montaut, Quentin Le Lidec, Antoine Bambade, et al. „Differentiable Collision Detection: a Randomized Smoothing Approach“. In: 2022. arXiv: `2209.09012 [cs.RO]` (cit. on p. 141).

[Mor+20] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. „Growing Neural Cellular Automata“. In: 2020 (cit. on p. 140).

[Mun14] Rémi Munos. „From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning“. In: vol. 7. 2014, pp. 1–129 (cit. on p. 27).

[Nau+20] Noel M. Naughton, Jiarui Sun, Arman Tekinalp, et al. „Elastica: A Compliant Mechanics Environment for Soft Robotic Control“. In: *IEEE Robotics and Automation Letters* 6 (2020), pp. 3389–3396 (cit. on p. 17).

[Nav+20] Stefan Escaida Navarro, Steven Nagels, Hosam Alagi, et al. „A Model-Based Sensor Fusion Approach for Force and Shape Estimation in Soft Robotics“. In: vol. 5. 2020, pp. 5621–5628 (cit. on pp. 13, 45).

[Nav+21] Stefan Navarro, Sepaldeep Dhaliwal, Mario Sanz-Lopez, et al. „A Bio-Inspired Active Prostate Phantom for Adaptive Interventions“. In: Oct. 2021 (cit. on p. 45).

[Nav+23] Stefan Escaida Navarro, Tanguy Navez, Olivier Goury, Luis Molina, and Christian Duriez. „An Open Source Design Optimization Toolbox Evaluated on a Soft Finger“. In: vol. 8. 9. Institute of Electrical and Electronics Engineers (IEEE), Sept. 2023, pp. 6044–6051 (cit. on pp. 114, 118, 140).

[Nic+18] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. „Gotta Learn Fast: A New Benchmark for Generalization in RL“. In: *arXiv preprint arXiv:1804.03720* (2018) (cit. on p. 20).

[NW06] J. Nocedal and S. J. Wright. „Numerical optimization“. In: Springer, 2nd ed., 2006 (cit. on p. 30).

[OGL13] David E. Orin, Ambarish Goswami, and Sung-Hee Lee. „Centroidal dynamics of a humanoid robot“. In: vol. 35. 2013, pp. 161–176 (cit. on p. 36).

[Ope+19a] OpenAI, Ilge Akkaya, Marcin Andrychowicz, et al. „Solving Rubik’s Cube with a Robot Hand“. en. In: *arXiv:1910.07113 [cs, stat]* (Oct. 2019). arXiv: 1910.07113 (cit. on p. 20).

[Ope+19b] OpenAI, Marcin Andrychowicz, Bowen Baker, et al. „Learning Dexterous In-Hand Manipulation“. en. In: *arXiv:1808.00177 [cs, stat]* (Jan. 2019). arXiv: 1808.00177 (cit. on pp. 20, 50).

[Pan+23] Tao Pang, H. J. Terry Suh, Lujie Yang, and Russ Tedrake. „Global Planning for Contact-Rich Manipulation via Local Smoothing of Quasi-dynamic Contact Models“. In: 2023. arXiv: 2206.10787 [cs.RO] (cit. on p. 80).

[Par20] J Parker-Holder. „Minimal pytorch soft actor critic (sac) implementation“. In: GitHub, 2020 (cit. on p. 84).

[Pat+21] Shubham Pateria, Budhitama Subagdja, Ah-Hwee Tan, and Chai Quek. „Hierarchical Reinforcement Learning: A Comprehensive Survey“. In: *ACM Computing Surveys* 54 (June 2021), pp. 1–35 (cit. on p. 25).

[PG15] Lerrel Pinto and Abhinav Gupta. „Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours“. en. In: *arXiv:1509.06825 [cs]* (Sept. 2015). arXiv: 1509.06825 (cit. on p. 20).

[PH22] Joshua Pinskier and David Howard. „From Bioinspiration to Computer Generation: Developments in Autonomous Soft Robot Design“. In: *Advanced Intelligent Systems* (2022) (cit. on p. 34).

[PR97] Ronald E. Parr and Stuart J. Russell. „Reinforcement Learning with Hierarchies of Machines“. In: *NIPS*. 1997 (cit. on p. 25).

[Put94] Martin L. Puterman. „Markov Decision Processes: Discrete Stochastic Dynamic Programming“. In: *Wiley Series in Probability and Statistics*. 1994 (cit. on p. 22).

[RDM19] Mark Runciman, Ara Darzi, and George P Mylonas. „Soft robotics in minimally invasive surgery“. In: *Soft robotics* 6.4 (2019), pp. 423–443 (cit. on p. 3).

[Ren+17] Federico Renda, Frédéric Boyer, Jorge Manuel Miranda Dias, and Lakmal D. Seneviratne. „Discrete Cosserat Approach for Multisection Soft Manipulator Dynamics“. In: *IEEE Transactions on Robotics* 34 (2017), pp. 1518–1533 (cit. on p. 15).

[RN94] Gavin Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*. Tech. rep. University of Cambridge, Department of Engineering Cambridge, UK, 1994 (cit. on p. 24).

[RNA22] Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. „Deep Generative Models in Engineering Design: A Review“. In: 2022. arXiv: 2110.10863 [cs.LG] (cit. on p. 140).

[Ros+19] Laura Ros-Freixedes, Anzhu Gao, Ning Liu, Mali Shen, and Guang-Zhong Yang. „Design optimization of a contact-aided continuum robot for endobronchial interventions based on anatomical constraints“. In: vol. 14. 2019, pp. 1137–1146 (cit. on p. 34).

[RT15] Daniela Rus and Michael T. Tolley. „Design, fabrication and control of soft robots“. In: *Nature* 521.7553 (2015), pp. 467–475 (cit. on p. 33).

[Rus+16] Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, et al. „Policy Distillation“. In: *CoRR* abs/1511.06295 (2016) (cit. on p. 25).

[Sch+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. „Proximal Policy Optimization Algorithms“. In: *ArXiv* abs/1707.06347 (2017) (cit. on p. 24).

[Sei+20] Daniel Seita, Pete Florence, Jonathan Tompson, et al. „Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks“. In: *arXiv preprint arXiv:2012.03385* (2020) (cit. on p. 17).

[SG05] Linda Smith and Michael Gasser. „The Development of Embodied Cognition: Six Lessons from Babies“. In: *Artificial Life* 11 (2005), pp. 13–29 (cit. on p. 5).

[SG19] Andrew N. Sloss and Steven Gustafson. *2019 Evolutionary Algorithms Review*. 2019 (cit. on p. 34).

[SGW15] Yoel Shapiro, Kosa Gabor, and Alon Wolf. „Modeling a hyperflexible planar bending actuator as an inextensible euler–bernoulli beam for use in flexible robots“. In: *Soft Robotics* 2.2 (2015), pp. 71–79 (cit. on pp. 15, 68).

[Sha+16] Bobak Shahriari, Kevin Swersky, Ziyun Wang, Ryan P. Adams, and Nando de Freitas. „Taking the Human Out of the Loop: A Review of Bayesian Optimization“. In: vol. 104. 2016, pp. 148–175 (cit. on p. 68).

[She+11] Robert Shepherd, Filip Ilievski, Wonjae Choi, et al. „Multigait Soft Robot“. In: *Proceedings of the National Academy of Sciences of the United States of America* 108 (Nov. 2011), pp. 20400–3 (cit. on pp. 4, 48).

[Sig97] Ole Sigmund. „On the design of compliant mechanisms using topology optimization“. In: *J. Struct. Mech.* 25.4 (1997), pp. 493–524 (cit. on p. 34).

[Sin+19] Nina R. Sinatra, Clark B. Teeple, Daniel M. Vogt, et al. „Ultragentle manipulation of delicate structures using a soft robotic gripper“. In: vol. 4. 33. 2019, eaax5425. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.aax5425` (cit. on p. 3).

[SIT91] Koichi Suzumori, Shoichi Iikura, and Hirohisa Tanaka. „Development of flexible microactuator and its applications to robotic mechanisms“. In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation* (1991), 1622–1627 vol.2 (cit. on pp. 3, 4).

[SK22] Filippo A. Spinelli and Robert K. Katzschmann. „A Unified and Modular Model Predictive Control Framework for Soft Continuum Manipulators under Internal and External Constraints“. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2022 (cit. on pp. 30, 80).

[SP08] A. J. Shaiju and Ian R. Petersen. „Formulas for Discrete Time LQR, LQG, LEQG and Minimax LQG Optimal Control Problems“. In: vol. 41. 2008, pp. 8773–8778 (cit. on p. 28).

[Spi+19] Andrew Spielberg, Allan Zhao, Tao Du, and al. „Learn- ing-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations“. In: *roc. Adv. Neural Inf. Process. Syst.* (2019), pp. 8284–8294 (cit. on p. 34).

[SPS99] Richard S. Sutton, Doina Precup, and Satinder Singh. „Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning“. In: *Artificial Intelligence* 112.1 (1999), pp. 181–211 (cit. on pp. 23, 25).

[SSW22] Charles Schaff, Audrey Sedal, and Matthew R. Walter. „Soft Robots Learn to Crawl: Jointly Optimizing Design and Control with Sim-to-Real Transfer“. In: (2022). arXiv: `2202.04575 [cs.RO]` (cit. on pp. 3, 4, 35).

[Sud+21] Shyam Sudhakaran, Djordje Grbic, Siyan Li, et al. „Growing 3D Artefacts and Functional Machines with Neural Cellular Automata“. In: 2021. arXiv: `2103.08737 [cs.LG]` (cit. on p. 140).

[Sui+19] Xin Sui, Hegao Cai, Dongyang Bie, et al. „Automatic Generation of Locomotion Patterns for Soft Modular Reconfigurable Robots“. In: *Applied Sciences* 10.1 (Dec. 2019), p. 294 (cit. on p. 34).

[Tag+20] Eleonora Tagliabue, Ameya Pore, Diego Dall’Alba, et al. „Soft tissue simulation environment to learn manipulation tasks in autonomous robotic surgery“. In: *2020 IEEE International Conference on Intelligent Robots and Systems (IROS). IEEE* (2020) (cit. on p. 17).

[Tal+15] Anthony Talvas, Maud Marchal, Christian Duriez, and Miguel A. Otaduy. „Aggregate Constraints for Virtual Manipulation with Soft Fingers“. In: vol. 21. 2015, pp. 452–461 (cit. on p. 19).

[Tea19] Team DEFROST (INRIA/CRISTAL) Lille. *SoftRobots plugin for SOFA*. `https://github.com/SofaDefrost/SoftRobots`. 2019 (cit. on p. 17).

[Tes95] Gerald Tesauro. „Temporal difference learning and TD-Gammon“. In: *Commun. ACM* 38 (1995), pp. 58–68 (cit. on p. 24).

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. „Mujoco: A physics engine for model-based control“. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 5026–5033 (cit. on p. 16).

[Tha94] Chen Khong Tham. „Modular on-line function approximation for scaling up reinforcement learning“. PhD thesis. University of Cambridge, 1994 (cit. on p. 24).

[Thu+17] Thomas George Thuruthel, Egidio Falotico, Mariangela Manti, et al. „Learning Closed Loop Kinematic Controllers for Continuum Manipulators in Unstructured Environments.“ In: *Soft robotics* 4 3 (2017), pp. 285–296 (cit. on p. 15).

[Thu+19] Thomas George Thuruthel, Egidio Falotico, Federico Renda, and Cecilia Laschi. „Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators“. In: *IEEE Transactions on Robotics* 35 (2019), pp. 124–134 (cit. on p. 15).

[TLP20] Sander Tonkens, Joseph Lorenzetti, and Marco Pavone. „Soft Robot Optimal Control Via Reduced Order Finite Element Models“. In: 2020, pp. 12010–12016 (cit. on p. 31).

[Tob+17] Joshua Tobin, Rachel Fong, Alex Ray, et al. „Domain randomization for transferring deep neural networks from simulation to the real world“. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), pp. 23–30 (cit. on p. 25).

[Tri+08] Deepak Trivedi, Christopher D. Rahn, William M. Kier, and Ian D. Walker. „Soft robotics: Biological inspiration, state of the art, and future research“. In: *Applied Bionics and Biomechanics* 5 (2008), pp. 99–117 (cit. on p. 14).

[UA22] Wyatt Ubellacker and Aaron Ames. „Robust Locomotion on Legged Robots through Planning on Motion Primitive Graphs“. In: 2022. arXiv: `2209.07503 [cs.RO]` (cit. on p. 80).

[Van+20] Félix Vanneste, Olivier Goury, Jonàs Martínez, et al. „Anisotropic Soft Robots Based on 3D Printed Meso-Structured Materials: Design, Modeling by Homogenization and Simulation“. In: vol. 5. 2020, pp. 2380–2386 (cit. on pp. 47, 66).

[VAP20] Filipe Veiga, Riad Akrour, and Jan Peters. „Hierarchical Tactile-Based Control Decomposition of Dexterous In-Hand Manipulation Tasks“. In: *Frontiers in Robotics and AI* 7 (2020) (cit. on p. 25).

[Vas+17] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, et al. „Attention is All you Need“. In: *NIPS*. 2017 (cit. on pp. 83, 124).

[Wan+16] Hongqiang Wang, Jie Chen, Henry Ying Kei Lau, and Hongliang Ren. „Motion Planning Based on Learning From Demonstration for Multiple-Segment Flexible Soft Robots Actuated by Electroactive Polymers“. In: *IEEE Robotics and Automation Letters* 1 (2016), pp. 391–398 (cit. on p. 20).

[Wan+20] Rixin Wang, Xianmin Zhang, Benliang Zhu, et al. „Topology optimization of a cable-driven soft robotic gripper“. In: *Structural and Multidisciplinary Optimization* 62.5 (2020), pp. 2749–2763 (cit. on p. 34).

[Wil+22] Jared D. Willard, Xiaowei Jia, Shaoming Xu, Michael S. Steinbach, and Vipin Kumar. „Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems“. In: *ACM Computing Surveys (CSUR)* (2022) (cit. on pp. 20, 24).

[WJ10] Robert J. Webster and Bryan A. Jones. „Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review“. In: *The International Journal of Robotics Research* 29 (2010), pp. 1661–1683 (cit. on pp. 14, 15, 67).

[WLK21] Xiaomei Wang, Yingqi Li, and Ka-Wai Kwok. „A Survey for Machine Learning-Based Control of Continuum Robots“. In: *Frontiers in Robotics and AI* 8 (2021) (cit. on p. 20).

[Wu+19] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. „Learning to manipulate deformable objects without demonstrations“. In: *arXiv preprint arXiv:1910.13439* (2019) (cit. on p. 17).

[Yar+21] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. „Reinforcement Learning with Prototypical Representations“. In: *ICML*. 2021 (cit. on p. 25).

[YHM23] Yao Yao, Liang He, and Perla Maiolino. „SPADA: A Toolbox of Designing Soft Pneumatic Actuators for Shape Matching based on Surrogate Modeling“. In: 2023. arXiv: `2305.19509 [cs.RO]` (cit. on p. 114).

[YZK20] Liu Yang, Dongkun Zhang, and George Em Karniadakis. „Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations“. In: *ArXiv* abs/1811.02033 (2020) (cit. on p. 24).

[Zha+15] Xiong Zhang, Zhen Chen, Jianhui Liao, and Yan Liu. „The material point method“. In: 2015 (cit. on p. 16).

[Zha+17] Hongying Zhang, Michael Yu Wang, Feifei Chen, et al. „Design and development of a soft gripper with topology optimization“. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 6239–6244 (cit. on p. 34).

List of Figures

List of Tables

Declaration

I declare here that I have completed my work solely and only with the help of the references I mentioned and the advice of my supervisor and colleagues.

Lille, December 21st, 2023

Etienne Ménager