Gait Adaptation and Iterative Control: a Switched Systems Optimization Framework for Quadrupedal Robots

Pietro Gori[1] *Student, IEEE*, Michele Pierallini[1] *Student, IEEE*, Franco Angelini[1] *Member, IEEE*, and Manolo Garabini[1] *Member, IEEE*

***Abstract*—One of the primary challenges in quadrupedal locomotion pertains to the robot's ability to adapt its gait to the surrounding environment and the desired task. This capability allows quadrupedal robots to select suitable foothold locations and adjust their gait for optimal performance. We address the problem of gait adaptation using Trajectory Optimization, which takes into account the simplified switched system's dynamics and optimizes the different phases of motion in which we split the robot's movement. The robot dynamic model is a Single Rigid Body with a rigid contact model and foot positions. We apply contact and friction cone constraints to ensure a physically feasible motion of the real robot. We tackle the optimization using the Direct Multiple Shooting method. Leveraging kinematic inversion to map the base and feet positions into joint positions, velocities, and accelerations, we design a controller that combines Iterative Learning Control and Proportional Derivative feedback control. The iterative controller compensates for the sim-to-real gap, allowing the real robot to learn the task during the execution of the latter. We evaluate the performance of the proposed approach on two different quadrupedal robots and on different terrains.**

***Index Terms*—Legged robots, gait generation, motion phases, Trajectory Optimization, planning with contacts, switched dynamics, Optimal Control, Iterative Learning Control.**

I. Introduction

QUADRUPEDAL robots are versatile and agile platforms suitable for operating in unstructured environments. These robots are inspired by their animal counterpart such as dogs, cats, and horses, whose locomotion is efficient and adaptable to traverse different types of terrain. Biologists have classified animal gaits and postures to understand better their motion [1], thus understanding why animals change gaits and how it ties to their speed is crucial. Energy consumption is the primary factor in these discussions. Bio-mechanic researchers have indicated that quadrupeds should transition from a walk to a trot to a gallop to reduce their energy expenditure and improve efficiency [2].

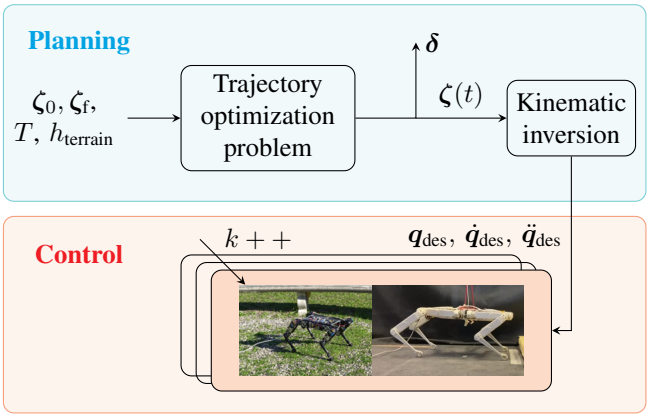


Figure 1: Pipeline overview. The figure illustrates the two-stage framework for planning and controlling the quadrupedal robot movement.

Quadrupedal robots have the advantage of balancing agility, which is higher than that of wheeled robots but lower than that of drones, with autonomy, which is higher than that of drones but lower than that of wheeled robots [3]. These robots often have sufficient carrying capacity for goods needed for inspection activities [4] as well as other drones and rescue equipment [5]. One of the key challenges in the locomotion of quadrupedal robots is that, in most cases, they exploit a fixed gait while moving. As a consequence, they can not adjust their footholds to climb steps or avoid obstacles.

This research aims to enhance the autonomy of quadrupedal robots while finding the best gait sequence for performing a task. Trajectory Optimization (TO) [6], [7] is an approach typically used to get optimal behaviors for quadrupedal robots. Moreover, we leverage a control framework based on Iterative Learning Control (ILC) and Proportional Derivative (PD) feedback controller to track the joint references and compensate for the reality gap [8].

A. Related Work

In recent years, researchers have focused on understanding how to automatize the process of foothold selection and gait discovery in quadrupedal robots. Another challenge is

This research is partially supported by the Italian Ministry of University and Research (MUR) - Fondo Italiano per la Scienza Applicata (FISA), through the OCCAM project under Grant FISA 2023-00324 CUP I53C25000700001 and the FoReLab" (Future-oriented Research Lab) Project (Departments ofâExcellence), and partially by the European Union HORIZON-MSCA-2023-SE-01-01 - MSCA Staff Exchanges 2023 Program under the Grant Agreement No.101182891 (NEUTRAWEED).
Editorial version published at https://ieeexplore.ieee.org/abstract/document/11159153.

Pietro Gori, Michele Pierallini, Franco Angelini, and Manolo Garabini are with the [1]Centro di Ricerca 'Enrico Piaggio" and the Dipartimento di Ingegneria dell'Informazione, Universita' di Pisa, Largo Lucio Lazzarino 1, 56126 Pisa, Italy (email: {pietro.gori, michele.pierallini}@phd.unipi.it and {franco.angelini, manolo.garabini}@unipi.it).

to discover a gait while finding a feasible movement for quadrupedal robots, which is achieved through contact-implicit planning. In literature, various works deal with contact-implicit methods, such as [9], [10], [11], [12], [13], [14]. In [9], a novel method for TO, which embeds the complementarity constraints formulation, is presented. This method defines the orthogonal relationship between contact forces and foot position, together with the non-penetration condition. Additionally, they do not consider an a priori mode ordering. In [10], the authors tackle TO by framing it as an optimal control problem for switched systems, leveraging the theoretical foundations of optimal control for such systems [15], [16]. A pivotal contribution of their work is the development of a continuous-time constrained Sequential Linear Quadratic (SLQ) algorithm [11], which efficiently handles state and input equality constraints with linear time complexity. This approach is employed in the OCS2 framework, a TO method that extends beyond traditional techniques by optimizing the entire system trajectory while maintaining a fixed mode sequence. This represents a significant advancement over earlier methods that primarily focused on control actions and switching times [16]. As in [9], they consider a contact-implicit method that relies on a complementarity formulation of the contact dynamics. Moreover, they assume that the gait sequence is predefined, which is a restriction in finding the absolute optimal behavior of the robot. The Authors in [12] use a formulation that relies on a smooth contact model, while a gait-free whole-body TO approach is performed. However, not all the optimized motions and gaits can be transferred to physical systems due to either the limitations of the real hardware or the deterministic nature of their TO, which does not provide robustness considerations. [13] provides a different approach because it exploits two optimization stages. The first one considers a continuous dynamic model, which is simpler than the full dynamics, with a smoothed contact model, to find the optimal gait. In the second stage, full hybrid dynamics is used to obtain a feasible robot's motion that can handle the gait found in the first stage. In contrast to the previous works, [14] addresses a contact-implicit TO while considering rigid contacts. Nevertheless, the presented method is highly sensitive to the initial guess. As a result, it may not be able to find contact sequences that differ significantly from the initial guess.

The problem of gait discovery is also addressed using Mixed-Integer Programming (MIP). However, MIPs optimize over the contact sequence, without considering the contact dynamics in the optimization problem [17]. In the worst case, they are not numerically efficient, because they must explore all the possible mode sequences to find the optimal solution.

As introduced in [13], researchers are currently emphasizing the use of simple models to plan the movement of complex robots. Simplifying the dynamics makes it possible to solve quickly hard numerical problems, such as Nonlinear Programming (NLP). This idea is captured in [18], [19], [20]. [18] presents a single TO formulation where the gait sequence, the footholds, and the body motion are optimized. In contrast to other works, the Authors solve an NLP enforcing the dynamics of a Single Rigid Body (SRB) to get the motion of the Center of Mass (CoM) of the robot. Additionally, foot positions are optimized to get the optimal footholds, while the optimization of the gait sequence is obtained considering the duration of each foot configuration, i.e., contact or swing. However, the optimal solution reached is not always feasible for a real robot. This work uses the software TOWR as a solver, which is specifically made to optimize the motion. In [19], an SRB model is used to perform the TO over a bipedal robot, in conjunction with hybrid systems theory. The paper relies on sim-to-real reinforcement learning to transfer the SRB reference trajectories to a full-order robot model and hardware. However, there are still challenges in crossing the sim-to-real gap, especially for high-speed gaits and more dynamic behaviors. Other simplified models are the Spring Linear Inverted Pendulum (SLIP) [21], [20] and the Soft Bilinear Inverted Pendulum (SBIP) [22]. [20] uses a version of the SLIP, called the VHSLIP model. Because of its inherent suitability for mitigating collisions with the terrain, the VHSLIP model accurately represents the robot's linear dynamics. Moreover, [22] tackles the problem of planning with a simplified model in the case of compliant terrain or robots equipped with soft feet.

The problem of gait and TO involving a simplified model and a rigid contact model is a recurring problem in the literature [18], [23], [17], [24]. As discussed in [25], researchers have tackled this problem in a Direct Collocation fashion. Nevertheless, to the best of the authors' knowledge, there are no relevant works that solve the same problem using the Direct Multiple Shooting (DMS) method [26], [27].

Once the gait is selected, the simplified model leads to errors and mismatches in the execution of the real motion. To compensate for the mismatches, machine learning approaches operate by randomizing the environment variables [19], conversely, model-based techniques take advantage of anticipatory high gains feedback loops [10]. ILC strategies belong at the intersection of the two worlds, they leverage data to improve the tracking performance generating a pure feedforward control action. In literature, there are several works about ILC, such as [28], [29], [30], [31]. [28], [29], [30] address the problem of method generalization. They generalize to new tasks starting from some significant knowledge, namely, the speed of trajectory execution or the model. However, [28] does not generalize for different trajectories, while [29] leverages pure feedforward control action without position and velocity tracking. [30], [31] introduce methods to allow the learning despite changes in the initial conditions iteration per iteration. Particularly, [31] leverages neural networks to achieve those results. Nevertheless, in the control stage, we introduce and demonstrate an ILC based on switched systems that requires a smaller amount of data than the one used for the NN training.

On quadrupedal robots, [32] introduces a novel ILC formulation for nonlinear robust jumping control, enabling fast learning of optimal actions while mitigating the need for a full-body dynamic model. Additionally, they are the first to combine TO and ILC for legged robots, forging a new approach to robust jumping control. However, in their planning strategy, the robot's motion is optimized with a fixed gait, whereas our pipeline allows optimizing both gait and motion. The ILC primarily aims to enhance robustness and optimize

reference tracking for explosive tasks. Notably, the objective of ILC does not encompass the training process of acquiring locomotor skills.

B. Contributions

Our research develops a control pipeline for quadruped locomotion using switched systems theory, comprising planning and control phases. The planning phase optimizes gait sequence, footholds, and body motion via TO, enforcing friction constraints and mapping results to joint references using kinematic inversion [33]. The control phase integrates ILC [34] and PD feedback for tracking. Experiments on quadrupedal robots [35], [36] showcase flexibility. Contributions include:

1) Optimizing phase durations for gait optimization via switched systems.
2) Using DMS [26], [27] for efficient NLP solving.
3) Deploying ILC to address sim-to-real gaps and improve performance.

II. Trajectory Optimization Formulation for Switched Systems

Our framework addresses a TO problem exploiting the switched systems theory. A switched system [37], [38], [39] is composed of multiple modes, i.e., subsystems, and a switching law that determines the active mode at any given time. Defining $\mathcal{I}$ as the set of indices of modes and $N_\mathrm{I} \in \mathbb{N}$ as the total number of modes, a switched system is defined as [39]

$$\begin{aligned} &\dot{\boldsymbol{\zeta}}(t) = \boldsymbol{f}_i(\boldsymbol{\zeta}(t)) + \boldsymbol{g}_i(\boldsymbol{\zeta}(t))\boldsymbol{u}(t) \\ &\text{with} \quad i \in \mathcal{I} \triangleq \{1, \dots, N_\mathrm{I}\}, \end{aligned} \tag{1}$$

where $\boldsymbol{\zeta}(t) \in \mathbb{R}^n$ is the state, $\boldsymbol{u}(t) \in U_{(\cdot)} \subseteq \mathbb{R}^m$ is the control input, and $\boldsymbol{f}_i(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$, $\boldsymbol{g}_i(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are the continuous function that represent the generic active mode i. The sequence of active modes is regulated by the switching sequence, i.e., $\sigma(\cdot) : [0, T] \to \mathcal{I}$, where $0, T$ represent the initial and final times, respectively. Since $\sigma(t)$ is a piecewise constant function, we suppose that it contains a finite number of discontinuities. These discontinuities are known as switching instants, and they represent the times when the system switches between modes. We define with $\boldsymbol{\tau} = [\tau_1, \tau_2, \dots, \tau_{N_\mathrm{T}}]^\top \in \mathbb{R}^{N_\mathrm{T}}$ the sequence of the switching instants, where N_T indicates the total number of switching. We denote with i_k that mode $(\boldsymbol{f}_i, \boldsymbol{g}_i)$ stays active across the time span $[t_k, t_{k+1}]$, where i represents the generic mode and k is the time instant when the transition happens.

Nevertheless, our approach does not directly consider the switching instants. Instead, we focus on the duration of each active mode. We define the duration of each active mode., i.e., δ, such as $\delta_k = \tau_{k+1} - \tau_k$ and

$$\Delta(T) \triangleq \left\{ \boldsymbol{\delta} \in \mathbb{R}^{N_\mathrm{T}+1} \,|\, \delta_i \geq 0 \,\wedge\, \sum_{i=0}^{N_\mathrm{T}} \delta_i = T \right\}, \tag{2}$$

where δ_k is the kth interval in which the time horizon of duration T is divided.

The choice of considering the switched systems theory is because, with (2), we optimize over modes duration when solving an Optimal Control Problem (OCP). Thus, to address a TO problem, we solve an OCP that fulfills (1), and therefore we can optimize the state trajectory, the control inputs, and the modes' duration.

The objective function in (3a) consists of two parts: an integral cost contribution that depends on both the state and controls, and a terminal cost depending on the distance between the final state, i.e., $\boldsymbol{\zeta}(t_\mathrm{f})$, and the desired state, i.e., $\boldsymbol{\zeta}_\mathrm{f}$. The initial state value is enforced through (3c). We enforce (3d) to find a feasible optimal solution. Finally, we solve the OCP as

$$\min_{\boldsymbol{\zeta},\, \boldsymbol{u},\, \boldsymbol{\delta}} \quad J(\boldsymbol{\zeta}(t),\, \boldsymbol{u}(t)) \tag{3a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{\zeta}}(t,\, \boldsymbol{\delta}) = \boldsymbol{f}_{\sigma(t)}(\boldsymbol{\zeta}(t)) + \boldsymbol{g}_{\sigma(t)}(\boldsymbol{\zeta}(t))\boldsymbol{u}(t)),\, t \in [0,\, T]\,, \tag{3b}$$

$$\boldsymbol{\zeta}(t_0) = \boldsymbol{\zeta}_0, \tag{3c}$$

$$\boldsymbol{h}(\boldsymbol{\zeta}(t), \boldsymbol{u}(t)) \leq 0,\, t \in [0,\, T]\,, \tag{3d}$$

$$\boldsymbol{u} \in U, \tag{3e}$$

$$\boldsymbol{\delta} \in \Delta(T). \tag{3f}$$

Our purpose is to represent the dynamic behavior of a quadrupedal robot through a switched system, according to (1), to solve (3).

In Sec. II-A to II-F, we explain how we address the OCP. After, Sec. III implements a controller framework to ensure that the robot follows the planned trajectory. The overall pipeline is depicted in Fig. 1.

A. Robot Model

The dynamic behavior of the robot is modeled through a simplified model, instead of the full dynamics. As introduced in Sec. I-A, the use of a 2D SRB model enables a significant reduction in computational complexity during TO, making it feasible to generate locomotion plans efficiently. While this simplification does not capture the full 3D dynamics, it allows for rapid planning and explicit modeling of the system's switching dynamics. The limitations of the 2D model are compensated by a robust data-driven control strategy, ensuring that the overall approach remains effective for the full 3D robot. This balance between model simplicity and control robustness is a key contribution of the work. The 2D representation is depicted in Fig. 2. Recalling (1), we define the switched SRB model as follows:

$$\begin{aligned} &\ddot{x} = \frac{1}{m}\sum_{j\in\mathcal{F}_i} F_j^x, \qquad \ddot{z} = \frac{1}{m}\sum_{j\in\mathcal{F}_i} F_j^z - g \\ &\ddot{\theta} = \frac{1}{I_z}\sum_{j\in\mathcal{F}_i} \boldsymbol{F}_j \times (\boldsymbol{p}_j - \boldsymbol{r}), \end{aligned} \tag{4}$$

where $m, I_z \in \mathbb{R}$ refer to the mass and the inertia of the robot along Z axis, respectively.

Recalling (1), the subset $\mathcal{F}_i$ defines which feet are in contact with the ground in each mode i. $\boldsymbol{r} = [x, z]^\top$ denotes the position of the CoM in the Cartesian plane, along with its derivatives $\dot{\boldsymbol{r}}$, $\ddot{\boldsymbol{r}}$, while θ is the pitch angle of the robot and $\dot{\theta}$ is its derivative. g is the gravity acceleration. There are also two additional terms for each foot: $\boldsymbol{F}_j(t) = \left[F_j^x, F_j^z\right]^\top \in \mathbb{R}^2$ and $\boldsymbol{p}_j(t) = \left[p_j^x, p_j^z\right]^\top \in \mathbb{R}^2$ representing contact forces and foot positions, respectively.

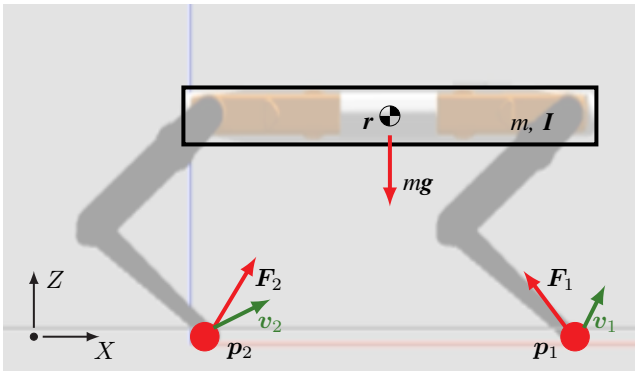


Figure 2: Representation of the 2D SRB model. In the background, the robot is displayed. The TO takes into account just the CoM position, i.e., $\boldsymbol{r}(t)$, and the feet positions, i.e., $\boldsymbol{p}_i(t)$. As optimization inputs, we consider the contact forces, i.e., $\boldsymbol{F}_i(t)$, and the feet velocities, i.e., $\boldsymbol{v}_i(t)$, with $i \in \{1, 2\}$. The motion along the Y direction is neglected in the trajectory optimization stage.

Equation (4) describes the CoM robot's motion. We augment our model by including the kinematic behavior of feet position, i.e., $\boldsymbol{p}_j$. The equations by which we describe each foot's kinematics are

$$\begin{aligned} \dot{p_j^x} &= v_j^x + \dot{x} + \dot{\theta}(p_j^z - z) \\ \dot{p_j^z} &= v_j^z + \dot{z} - \dot{\theta}(p_j^x - x), \end{aligned} \tag{5}$$

where $\left[v_j^x, v_j^z\right] \in \mathbb{R}^2$ are velocity controls, with $j \in \mathcal{F}_i$, and the remaining part of the equations represents the influence of the CoM on the feet. The TO problem returns the optimal motions of both the CoM (4) and the feet (5). Our model is still simplified, but it maximizes the information considered in the Cartesian space without implementing a full dynamics model.

B. Motion Phases Description

A key point in solving (3) is how we consider the switching sequence, i.e., $\sigma(t)$. In contrast with [10], [15], [16], we do not optimize over the switching sequence, thus we do not consider optimization strategies proposed in those works. We do not even consider a MIP strategy to optimize contacts, because we include the contact dynamics in the framework. Our focus is the optimization of the active modes' duration [40], also called phase sequence or switching sequence.

We note that, according to (2), successive switching times can be equal, i.e., $\delta_k = 0 \implies \tau_k = \tau_{k+1}$. This indicates that the switching sequence will change passing from $(i-1)_{k-1}$ to $(i+1)_{k+1}$. Consequently, the switching sequence is altered by changing the continuous variable δ_i, rather than directly varying the integer value i in (1).

For the 2D system (4), we assume that the front legs move simultaneously, and the same goes for the back legs, i.e., pronking and bounding gaits can be generated. This allows for consideration of only one contact point for the front feet and another for the rear feet.

In this manuscript, we represent a single step of the robot through a specific motion sequence. This results in the definition of a finite number of modes that build the switched system defined in (1). We organize the mode sequence of a step as follows: first, there is the stance phase, while both front and rear limbs are in contact with the ground. The following is the push phase, where only the rear limbs are in contact. Then, we call the flight phase the mode in which the robot has no contact with the ground. The last phase of the step, during which the quadruped touches the ground with only its front legs, involves the recover from the current step to the next step, so we call it the recover phase. This process is then repeated for a pre-specified number of steps, i.e., N_{step}. The main idea behind switched systems is to allow the robot to perform a feasible movement despite the removal of one or more phases as a result of the optimization process.

Through the optimization of the duration of the active mode [40], we optimize the behavior of the robot and its gait, which is not predefined because we provide only the initial guess for the phase sequence. The solver is free to adapt the value of each δ_k and find their optimal values to determine the robot's gait. It is important to note that the choice of the initial guess of the phase sequence can influence the gait.

C. Direct Multiple Shooting

The OCP discussed in Section II is addressed using a DMS method [26], [27]. This method is part of the direct methods family, and it aims to transcribe a continuous OCP into a finite NLP through discretization. DMS efficiently integrates dynamics over short time intervals. This results in a better optimization than Direct Single Shooting (DSS) [41] because, in DSS, the state becomes highly nonlinear during long integration periods. Furthermore, DMS optimizes over both control and state trajectory simultaneously.

Other approaches use the Direct Collocation method (DC) [18], [23], where the state trajectory is approximated with a polynomial between successive optimization instants. However, as evidenced by [25], there is a lack of significant research addressing the topic of TO on quadrupedal robots through DMS. This represents a significant challenge for this work.

Furthermore, there are cutting-edge algorithms, such as TOWR [18] and OCS2 [11], [10], that leverage different methods to address optimization problems, namely, DC and Sequential Linear Quadratic (SLQ). A comparison of DMS with SLQ reveals some notable differences. DMS is designed to solve nonlinear programming (NLP) problems, whereas SLQ is optimized for solving a sequence of linear quadratic (LQ) problems. However, the linearization process inherent to SLQ may introduce approximation errors. Additionally, DMS includes the systems's state in the optimization variables, whereas SLQ optimizes the state only by propagating the optimal control sequence.

DDP [42] is a state-of-the-art optimization technique that leverages Bellman's Optimality Principle to compute the state trajectory and the control input for a given predefined sequence of contacts. As SLQ, it optimizes the state only by propagating the optimal control sequence.

D. Optimization Variables

The state vector includes the linear and angular positions of the robot's CoM and their first derivatives: x, z, θ and $\dot{x}$, $\dot{z}$, $\dot{\theta}$. For the sake of clarity, we refer to the front and hind feet using the subscripts f and h, while in Sections II-A and II-F we use as a subscript $i \in \{1, 2\}$, with $n_\mathrm{f} = 2$. We augment the state vector with the linear positions of the front and hind feet: x_f, z_f, x_h, and z_h. At the end, we obtain

$$\boldsymbol{\zeta}(t) = [x, z, \theta, x_\mathrm{f}, z_\mathrm{f}, x_\mathrm{h}, z_\mathrm{h}, \dot{x}, \dot{z}, \dot{\theta}]^\top. \tag{6}$$

The controls are defined as

$$\boldsymbol{u}(t) = [F_{x_\mathrm{f}}, F_{z_\mathrm{f}}, F_{x_\mathrm{h}}, F_{z_\mathrm{h}}, v_{x_\mathrm{f}}, v_{z_\mathrm{f}}, v_{x_\mathrm{h}}, v_{z_\mathrm{h}}]^\top, \tag{7}$$

where F_{x_f}, F_{z_f}, F_{x_h} and F_{z_h} denote the contact forces acting along *X* and *Z* axis while v_{x_f}, v_{z_f}, v_{x_h} and v_{z_h} represent the velocity control utilized for the feet. Moreover, we consider the modes duration, i.e., $\boldsymbol{\delta}$, in the optimization vector.

E. Cost Function

In this paper, the cost function consists of different terms: the regulating cost $l_\mathrm{r}(\boldsymbol{u}(t))$, the tracking cost $l_\mathrm{t}(\boldsymbol{\zeta}(t))$, the clearance cost $l_\mathrm{cl}(\boldsymbol{\zeta}(t))$ and the final cost $l_N(\boldsymbol{\zeta}(T))$. The objective function J (3a), at a generic time instant t, is the following

$$\begin{aligned} &J(\boldsymbol{\zeta}(t),\, \boldsymbol{u}(t),\, \boldsymbol{\delta}) = l_\mathrm{r}(\boldsymbol{u}(t)) + l_\mathrm{t}(\boldsymbol{\zeta}(t)) + l_\mathrm{cl}(\boldsymbol{\zeta}(t)), \\ &J(\boldsymbol{\zeta}(T),\, \boldsymbol{u}(T),\, \boldsymbol{\delta}) = l_N(\boldsymbol{\zeta}(T)). \end{aligned} \tag{8}$$

$l_\mathrm{r}(\boldsymbol{u}(t))$ and $l_\mathrm{t}(\boldsymbol{\zeta}(t))$ are used in all the experiments, while $l_\mathrm{cl}(\boldsymbol{\zeta}(t))$ is specifically used for encouraging a well-defined motion.

1) Regulating Cost: The regulating cost is a quadratic term that weights the components of the control input. Specifically, it is designed to minimize the control input to reduce the effort of the robot in achieving its goal.

$$l_\mathrm{r}(\boldsymbol{u}(t)) = \int_0^T \|\boldsymbol{u}(t)\|_R^2 \mathrm{d}t, \tag{9}$$

where $R \in \mathbb{R}^{m \times m}$ is the square matrix that weights the input.

2) Tracking Cost: The tracking cost is a quadratic term that penalizes the distance of the state $\boldsymbol{\zeta}(t)$ from the desired state $\boldsymbol{\zeta}_\mathrm{d}$. This distance is also penalized in the final time cost.

$$\begin{aligned} &l_\mathrm{t}(\boldsymbol{\zeta}(t)) = \int_0^T \|\boldsymbol{\zeta}_\mathrm{d} - \boldsymbol{\zeta}(t)\|_Q^2 \mathrm{d}t, \\ &l_N(\boldsymbol{\zeta}(T)) = \|\boldsymbol{\zeta}_\mathrm{f} - \boldsymbol{\zeta}(T)\|_E^2, \end{aligned} \tag{10}$$

where $Q \in \mathbb{R}^{n \times n}$ and $E \in \mathbb{R}^{n \times n}$ are the weight matrices during the time horizon and at the final time, respectively.

3) Clearance Cost: No constraints are imposed on the robot to facilitate the discovery of a particular gait. However, we introduce a clearance cost term in the cost function that can assist the solver in finding a feasible gait. This result is achieved by forcing the robot to elevate its feet and avoiding foot crawling conditions. This approach becomes crucial when the tracking cost does not result in a reduction of the cost associated with lifting the foot higher. The formulation of the clearance cost is as follows.

$$l_\mathrm{cl}(\boldsymbol{\zeta}(t)) = \int_0^T k_1 \|z_\mathrm{f}(t) - z_1\|^2 + k2 \|z_\mathrm{h}(t) - z_2\|^2 \mathrm{d}t, \tag{11}$$

where z_1 and z_2 denote the max foot height for front and hind feet, respectively. Additionally, k_1 and k_2 are tuning parameters.

F. Feasibility Constraints

To obtain a physically feasible solution, we impose some constraints, which are represented in the OCP by (3d). Kinematic constraints are applied to link (4) and (5), relating the movements of the CoM and feet. The distance between each foot and its corresponding hip is explicitly restricted. This constraint is necessary since the hip position is rigidly linked to the CoM. We define the constraints as bounding boxes that measure l_box in width and h_box in height, both of which are user-defined parameters.

$$\begin{aligned} -l_\mathrm{box} \leq x_s(t) - x_i(t) \leq l_\mathrm{box} \\ d_\mathrm{min} \leq z_s(t) - z_i(t) \leq d_\mathrm{min} + h_\mathrm{box}, \end{aligned} \tag{12}$$

where $x_i(t)$ and $z_i(t)$ are the foot coordinates, with $i \in \{f, h\}$. $x_s(t)$ and $z_s(t)$ define the hip position in the world frame, while d_min is the minimum distance allowed between hip and foot on the *Z* axis, also user-defined.

Equation (3f) shows that the vector of modes duration, i.e., $\boldsymbol{\delta}$, has to fulfill the properties of the $\Delta(T)$ set defined in (2).

To determine if the robot is in contact with the ground, we define the $h_\mathrm{terrain}(x)$ function, which describes the shape of the terrain. To detect ground contact, the function is then compared to each foot's z position at each x location. If $z_{\mathrm{foot}_i} - h_\mathrm{terrain}(x_{\mathrm{foot}_i})$ falls below a specific threshold, then the foot is considered in contact with the ground, otherwise there is no contact. Additionally, other constraints are enforced to ensure obstacle avoidance. Moreover, we set force boundaries

$$\boldsymbol{F}_\mathrm{min} \leq \boldsymbol{F}_i(t) \leq \boldsymbol{F}_\mathrm{max} \quad \text{with } i \in \{f,\, h\},$$

where $\boldsymbol{F}_\mathrm{max}$, $\boldsymbol{F}_\mathrm{min} \in \mathbb{R}^2$ are the upper and lower force bounds respectively. Assuming that the friction coefficient μ has a standard value and denoting the contact forces, in a 2D scenario where the *Y* component is neglected, acting on a foot as $\boldsymbol{F}_i = [F_i^x, F_i^z]^\top$, we express the friction constraints as

$$\|F_i^x\| \leq \mu F_i^z \quad \text{with } i \in \{f,\, h\}. \tag{13}$$

We also set velocity boundaries to limit the feet' velocity as

$$\begin{aligned} v_\mathrm{min}^x \leq v_i^x(t) \leq v_\mathrm{max}^x \quad \text{with } i \in \{f,\, h\}, \\ v_\mathrm{min}^z \leq v_i^z(t) \leq v_\mathrm{max}^z \quad \text{with } i \in \{f,\, h\}, \end{aligned}$$

where v_max^x, v_max^z, v_min^x, and v_min^z are the maximum and minimum allowed velocities along *X* and *Z* directions, respectively.

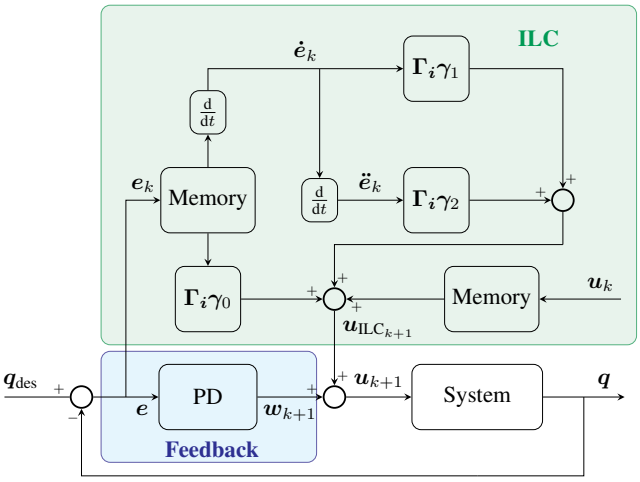


Figure 3: Control strategy overview. In the figure, are depicted the control law structure and the components involved in the ILC and PD feedback controller architecture. The diagram shows how the control inputs are updated and how the PD feedback is integrated with the ILC to enhance the robots performance in tracking the desired trajectory.

III. Control Framework

The TO problem, discussed in Section II, is incorporated into a framework used to plan the leg movements of a robot. As shown in Fig. 1, there are two main stages: planning and control. In the planning stage, given the initial state, the final desired state, the task duration, and the terrain profile, the TO calculates the optimal trajectories for the robot's base and feet. Additionally, it determines the optimal duration of each motion phase, i.e., Sec. II-B. After, we deploy kinematic inversion [33] to convert base and foot trajectories into the joint space. Finally, position references alone are insufficient for accurate trajectory tracking, necessitating velocity and acceleration references for the control stage. We obtain both joint velocities and accelerations through numerical offline differentiation. The contact forces from (7) were excluded from the controller due to performance degradation caused by mismatches between offline-optimized and actual forces, leading to noise and imperfect compensation.

The control framework, highlighted in red in Fig. 1, is responsible for ensuring that each joint follows its reference, i.e., $\boldsymbol{q}_{\mathrm{des}}$, $\dot{\boldsymbol{q}}_{\mathrm{des}}$, $\ddot{\boldsymbol{q}}_{\mathrm{des}} \in \mathbb{R}^{n_{\mathrm{j}}}$, thereby enabling the robot to move according to the planned trajectory. As illustrated in Fig. 1, the control framework is based on an ILC+PD feedback controller architecture, as introduced in [34], [8]. This control strategy is applied to each joint, facilitating the robot's learning of the task and enhancing its performance in terms of trajectory tracking while compensating for the unmodeled effect in the planning phase.

To enhance the robot's performance and compensate for the reality gap, we use ILC [34], a method that improves the trajectory tracking performance and the robustness of a system by exploiting repetitions of the desired task. Recalling Section I-A, our pipeline allows for TO even optimizing the robot's gait, using the switched system theory. This theory is also used in our control framework because the ILC method achieves trajectory tracking for the nonlinear switched system without complete knowledge of the robot's model [43]. Figure 3 illustrates that ILC updates a feed-forward law at each iteration, i.e., $\boldsymbol{u}_{\mathrm{ILC}_{k+1}}$, to refine the control input given to the system. We also deploy a PD feedback controller to track the joint references in a closed-loop fashion. This control strategy facilitates the desired behavior of the system, enhances the learning speed, and compensates for non-repeating disturbances, thereby enabling the robot to stand stably. Hence, the robot executes the planned motion. Our ILC method is designed for the nonlinear switched system class with arbitrary switching sequence [44], i.e., $\sigma(t)$. To guarantee the convergence of the ILC contribute, we extend [44, Theorem 1] demonstrating the convergence of the PD-type ILC law, instead of the D-type, considering an arbitrary switching sequence.

1) ILC Convergence: We here demonstrate the convergence of the ILC law for nonlinear switched systems. Let us define the state stacking positions and velocities of the robot's joints, i.e., $\boldsymbol{\zeta} = \left[\boldsymbol{q}^\top, \dot{\boldsymbol{q}}^\top\right]^\top \in \mathbb{R}^{2n_{\mathrm{j}}}$[1]. Thus, the dynamics can be written as (1) and its output function, i.e.,

$$\begin{cases} \dot{\boldsymbol{\zeta}}_k(t) = \boldsymbol{f}_i(\boldsymbol{\zeta}_k(t)) + \boldsymbol{g}_i(\boldsymbol{\zeta}_k(t))\boldsymbol{u}_k(t) \\ \boldsymbol{y}_k(t) = \boldsymbol{h}_i(\boldsymbol{\zeta}_k(t)) \\ i \in \mathcal{I} \end{cases} \tag{14}$$

where $\boldsymbol{h}_k(\cdot) : \mathbb{R}^{2n_{\mathrm{j}}} \to \mathbb{R}^{n_{\mathrm{j}}}$, with n_{j} is the number of actuated joints of the quadrupedal robot. Thus, we have the number of outputs equal to the number of inputs, i.e., a square system. Moreover, k denotes the system evolution at k-th iteration, and i is the active mode.

We assume for (14) what follows.

Assumption 1. The state $\boldsymbol{\zeta}(t) \in \mathbb{R}^n$ is continuous in the time domain, i.e., $t \in [0,\, T]$.

Assumption 2. The nonlinear function $\boldsymbol{f}_i(\cdot)$, $\boldsymbol{g}_i(\cdot)$, $\boldsymbol{h}_i(\cdot)$ [2], with $i \in \mathcal{I}$, are globally Lipschitz in $\boldsymbol{\zeta}(t)$ with $t \in [0,\, T]$, i.e., $\forall t \in [0,\, T]$, $\exists$ constants k_{f}, k_{g}, and k_{h} such that

$$\begin{aligned} \|\boldsymbol{f}_i(\boldsymbol{\zeta}_1(t)) - \boldsymbol{f}_i(\boldsymbol{\zeta}_2(t))\| &\leq k_{\mathrm{f}} \|\boldsymbol{\zeta}_1(t) - \boldsymbol{\zeta}_2(t)\| \\ \|\boldsymbol{g}_i(\boldsymbol{\zeta}_1(t)) - \boldsymbol{g}_i(\boldsymbol{\zeta}_2(t))\| &\leq k_{\mathrm{g}} \|\boldsymbol{\zeta}_1(t) - \boldsymbol{\zeta}_2(t)\| \\ \|\boldsymbol{h}_i(\boldsymbol{\zeta}_1(t)) - \boldsymbol{h}_i(\boldsymbol{\zeta}_2(t))\| &\leq k_{\mathrm{h}} \|\boldsymbol{\zeta}_1(t) - \boldsymbol{\zeta}_2(t)\| \end{aligned}$$

for any pair $(\boldsymbol{\zeta}_1(t),\, \boldsymbol{\zeta}_2(t))$.

Assumption 3. At each iteration, the following condition is satisfied

$$\boldsymbol{\zeta}_k(0) = \boldsymbol{\zeta}_{\mathrm{d}}(0),$$

where $\boldsymbol{\zeta}_{\mathrm{d}}(0)$ is the initial value of the desired trajectory.

Assumption 4. Given the desired state $\boldsymbol{\zeta}_{\mathrm{d}}(t)$, there exists a unique $\boldsymbol{u}_{\mathrm{d}}(t)$ such that

$$\begin{cases} \dot{\boldsymbol{\zeta}}_{\mathrm{d}}(t) = \boldsymbol{f}_i(\boldsymbol{\zeta}_{\mathrm{d}}(t)) + \boldsymbol{g}_i(\boldsymbol{\zeta}_{\mathrm{d}}(t))\boldsymbol{u}_{\mathrm{d}}(t) \\ \boldsymbol{y}_{\mathrm{d}}(t) = \boldsymbol{h}_i(\boldsymbol{\zeta}_{\mathrm{d}}(t)). \end{cases}$$

[1]Note that no information on the robot floating base is present in the state.

[2]For the sake of clarity, we introduce a compact notation to consider the partial derivatives and the various dependencies of the functions, i.e., $L_{f_i}\boldsymbol{h}_{i,\,k} \triangleq (\partial \boldsymbol{h}_i(\boldsymbol{\zeta}(t))/\partial \boldsymbol{\zeta})\, \boldsymbol{f}_i(\boldsymbol{\zeta}(t))|_{\boldsymbol{\zeta}=\boldsymbol{\zeta}_k(t)}$, $L_{g_i}\boldsymbol{h}_{i,\,k} \triangleq (\partial \boldsymbol{h}_i(\boldsymbol{\zeta}(t))/\partial \boldsymbol{\zeta})\, \boldsymbol{g}_i(\boldsymbol{\zeta}(t))|_{\boldsymbol{\zeta}=\boldsymbol{\zeta}_k(t)}$, $\boldsymbol{h}_{i,\,k} = \boldsymbol{h}_i(\boldsymbol{\zeta}_k(t))$, $\boldsymbol{h}_{i,\,\mathrm{d}} = \boldsymbol{h}_i(\boldsymbol{\zeta}_{\mathrm{d}}(t))$, $\boldsymbol{f}_{i,\,k} \triangleq \boldsymbol{f}_i(\boldsymbol{\zeta}_k(t))$, $\boldsymbol{f}_{i,\,\mathrm{d}} \triangleq \boldsymbol{f}_i(\boldsymbol{\zeta}_{\mathrm{d}}(t))$, $\boldsymbol{g}_{i,\,k} \triangleq \boldsymbol{g}_i(\boldsymbol{\zeta}_k(t))$, $\boldsymbol{g}_{i,\,\mathrm{d}} \triangleq \boldsymbol{g}_i(\boldsymbol{\zeta}_{\mathrm{d}}(t))$

Assumption 5. The system has relative degree r $(r > 0)$, $\forall n_\mathrm{j}$.

Assumption 1 is a standard condition for switched systems. It implies that the system does not have jumps, even on the switching time instants. The maximum Lipschitz constants for each of the subsystems are represented by the Lipschitz constants k_f, k_g, and k_h in Assumption 2. Assumption 3 is the typical initial condition used with ILC. The uniform convergence of the control profile $\boldsymbol{u}(t)$ to $\boldsymbol{u}_\mathrm{d}(t)$ indicates that the output tracking errors and the state tracking errors will vanish. This is based on the assertion in Assumption 4 that $\boldsymbol{u}_\mathrm{d}(t)$ exists uniquely. Assumption 5 allows us to generalize the method to a generic relative degree r, w.r.t. [44] which considers just $r = 1$. Finally, Assumptions 1-5 do not limit the applicability of the method.

The objective is to find a control input for (14) such that the output converges to the desired output, that is, $\boldsymbol{y}_k(t) \to \boldsymbol{y}_d(t)$ as $k \to \infty$. In this manuscript, we consider the ILC law as

$$\boldsymbol{u}_{ILC_{k+1}} = \boldsymbol{u}_k + \boldsymbol{\Gamma}_i \sum_{j=0}^{r} \boldsymbol{\gamma}_j \left(\boldsymbol{y}_\mathrm{d}^{(j)}(t) - \boldsymbol{y}_k^{(j)}(t) \right), \qquad (15)$$

where $\boldsymbol{e}_k(t) \triangleq \boldsymbol{y}_\mathrm{d}(t) - \boldsymbol{y}_k(t)$ is the tracking error at the $k-$th iteration and, recalling Fig. 3, $\boldsymbol{u}_k \in \mathbb{R}^{n_\mathrm{j}}$ represents the control action taken during the $k-$th trial. $\boldsymbol{\Gamma}_i \in \mathbb{R}^{n_\mathrm{j} \times n_\mathrm{j}}$ is the learning gain and $\boldsymbol{\gamma}_j \in \mathbb{R}^{n_\mathrm{j} \times n_\mathrm{j}}$, $\forall j = 0, \ldots, r$ are the tunable control weights, which affect the speed of the learning but not the convergence [45].

Theorem 1. Let us consider the nonlinear switched system (14), the switching rule $\sigma(t)$, and the ILC law (15). Under Assumptions 1-5, the output, i.e., $\boldsymbol{y}_k(t)$, converges to the desired output, i.e., $\boldsymbol{y}_\mathrm{d}(t)$, when $k \to \infty$ for all t, if

1) $\left\| I - \boldsymbol{\Gamma}_i L_{g_i} L_{f_i}^{(r-1)} \boldsymbol{h}_{i,k} \right\| \leq \rho < 1,\ 0 \leq t \leq T$
2) $\boldsymbol{\zeta}_k(0) = \boldsymbol{\zeta}_d(0)$

Proof. The proof is in Appendix A. □

For the sake of clarity, we consider the full dynamics of the robot as a nonlinear switched system, as in [10]. Thus, we choose $\boldsymbol{h}_k(\cdot)$ as the selection function that extracts only the joint positions, i.e., $\boldsymbol{y} \triangleq \boldsymbol{q} \in \mathbb{R}^{n_\mathrm{j}}$, from the entire state vector. Taking into account this information, we get that the relative degree r of our system is $r = 2$. The data and measurements of each joint's position, velocity, and acceleration are essential for proving Theorem 1. To obtain these values, it is necessary to estimate the acceleration measurements. To do so, we differentiate the velocity measurement after each trial. Then, we update the feed-forward law. We apply the differentiation offline. This method allows us to use more refined differentiation techniques, such as symmetrical numerical filters. The overall structure of the control law is depicted in Fig. 3. The following is the equation of the control law

$$\boldsymbol{u}_{k+1} = \underbrace{\boldsymbol{w}_{k+1}}_{\text{PD feedback}} + \underbrace{\boldsymbol{u}_{\mathrm{ILC}_{k+1}}}_{\text{feedforward}}, \qquad (16)$$

The term $\boldsymbol{w}_{k+1}$ represents the current PD feedback control action [8], whose law is

$$\boldsymbol{w}_{k+1} = \boldsymbol{K}_\mathrm{p} \boldsymbol{e}_{k+1} + \boldsymbol{K}_\mathrm{d} \dot{\boldsymbol{e}}_{k+1}, \qquad (17)$$

where $\boldsymbol{e}_{k+1}$, $\dot{\boldsymbol{e}}_{k+1} \in \mathbb{R}^{n_\mathrm{j}}$ are the current position and velocity joint errors, and $\boldsymbol{K}_\mathrm{p}$ and $\boldsymbol{K}_\mathrm{d} \in \mathbb{R}^{n_\mathrm{j} \times n_\mathrm{j}}$ are the positive-definite PD gain matrices. The $\boldsymbol{w}_{k+1}$ does not affect the convergence of the control method. The PD control gains can be tuned easily to reduce arbitrarily the tracking error and thus the control action, i.e., $\|\boldsymbol{w}_{k+1}\| \leq \boldsymbol{b}_{w_{k+1}} \in \mathbb{R}^{n_\mathrm{j}}$, as described in [46, Chapter 8.4]. Moreover, [47] demonstrates that it is possible to implement a closed-loop system to facilitate learning without encountering any significant issue. It is noteworthy that in practical applications, the Lipschitz conditions stated in Assumption 2 are typically satisfied as long as the robot operates within its nominal range of motion and avoids singular configurations.

IV. Results

The effectiveness of the proposed algorithm has been validated with seven experiments conducted using two different robots under different task conditions. The goal is to find a feasible motion and an appropriate gait to reach a desired position over a specific terrain.

The employed robots are Solo-12 [35] and Otto [36]. Solo-12 is a 12 Degrees of Freedom (DoF) quadrupedal robot with a body module and 4 identical 3 DoF legs. It is a lightweight robot, weighing only about $2.5\,\mathrm{kg}$. Otto is an 8 DoF quadrupedal robot with a body module and 4 identical 2 DoF legs. Its weight is about $5\,\mathrm{kg}$

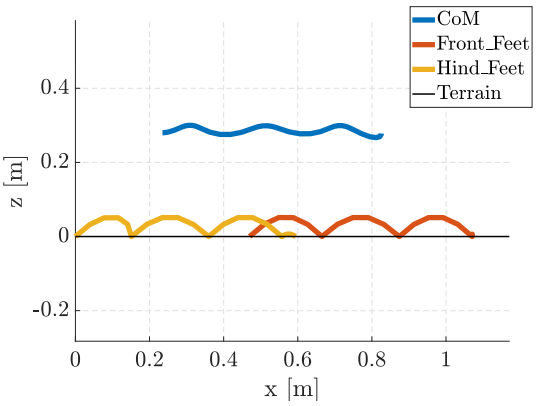
Two different terrains are tested: flat terrain (Sec. IV-A) using Otto, and a single-step terrain (Sec. IV-B) using both Otto and Solo-12. Given the differences between the two environments, the proposed planner outputs different gaits. The flat terrain planned trajectory is also tested under five different conditions to test the controller's robustness.

Our software is developed in Python with CasADi [48], an open-source library to solve optimization problems. We use `IPOPT` [49] as a solver. The TO, i.e., Sec. II, and kinematic inversion are performed offline on a laptop equipped with an Intel Core i7 processor and $32\,\mathrm{GB}$ of RAM. The planner parameters are specified in the Sec. IV-A and Sec. IV-B. While valuable methods like OCS2 prioritize faster online computation, our offline optimization approach emphasizes generating high-quality trajectories, which align better with the objectives of our framework.

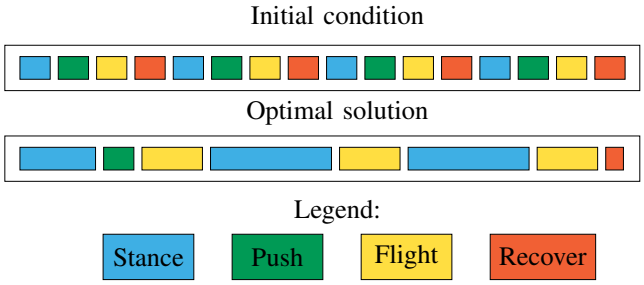
We provide only the joint position, velocity, and acceleration references to the robot controller. To fulfill and guarantee the convergence of the results presented in Theorem 1, we use the following gains: $\boldsymbol{\Gamma}_i = 0.001 I_{n_\mathrm{j} \times n_\mathrm{j}}$, $\boldsymbol{\gamma}_0 = 500 I_{n_\mathrm{j} \times n_\mathrm{j}}$, $\boldsymbol{\gamma_1} = 50 I_{n_\mathrm{j} \times n_\mathrm{j}}$, $\boldsymbol{\gamma_2} = I_{n_\mathrm{j} \times n_\mathrm{j}}$.

A. Experimental validation: Flat Terrain

In this section, we test the effectiveness of the proposed method by reaching the desired robot's position on a flat terrain using the Otto robot. First, the reference motion is planned given the specific environment. Then, to test the pipeline robustness, the same reference is experimentally validated under five different conditions: indoor terrain, indoor terrain with iteration-varying external disturbances, outdoor grass, outdoor gravel, and outdoor speed bump.



(a) CoM and feet trajectories obtained as a result of the TO problem.



(b) Optimal solution for the phase sequence on a flat terrain.

Figure 4: Planning results on flat terrain

The planning is computed using a task time horizon $T = 1\,\mathrm{s}$, while the total distance that the CoM has to cover is $0.6\,\mathrm{m}$ along the X axis, world frame. We set the optimization point number, i.e., N, equal to 80. Given the desired distance and the robot size, we estimate four steps. For this reason, we set the initial guess for the number of phases equal to 16 ($N_\mathrm{T} = 15$), and we set them a sequence of four times the four phases with equal duration ($T/(N_\mathrm{T}+1)$).

Figure 4 depicts the results of the TO problem on flat terrain. The planning outputs a three-step motion to reach the desired goal position, as evidenced in Fig. 4a. Moreover, Fig. 4b provides a phase description highlighting the difference between the initial guess and the optimal phase sequence. The robot' starting position is with CoM at $(0.23\,,0.28\,)\,\mathrm{m}$, while the rear and front feet are positioned at $(0\,,0\,)\,\mathrm{m}$ and $(0.46\,,0\,)\,\mathrm{m}$ in the world frame, respectively. The robot then moves to bring its CoM to the goal point: $(0.83\,,0.28\,)\,\mathrm{m}$.

The actual duration of each motion phase is displayed in Fig. 5. Upon examination of the plot, the TO problem found the pronking to be the optimal gait sequence. However, in the initial phase, the robot lifts the front feet before the hind feet. This transition signifies the progression from the stance phase to the push phase and subsequently to the flight phase, and, throughout the second and third steps, the front and hind feet exhibit a synchronized movement. In the final part of the task, it is observed that only the hind feet are lifted, thereby stabilizing the robot's final position.

For each tested condition, 15 ILC iterations were performed. Figure 6 presents a comparison across the five experiments of the Root Mean Square (RMS) joint position error over

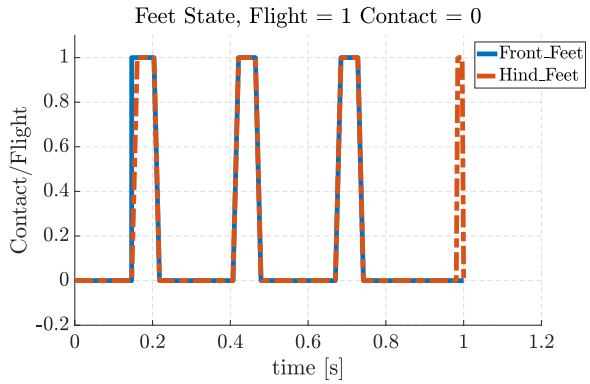


Figure 5: Feet state during task execution. It highlights the precise timing when the robots feet are stationary or in motion, reflecting the optimization results for efficient locomotion. The figure emphasizes the importance of timing in coordinating the robots gait for successful task completion.

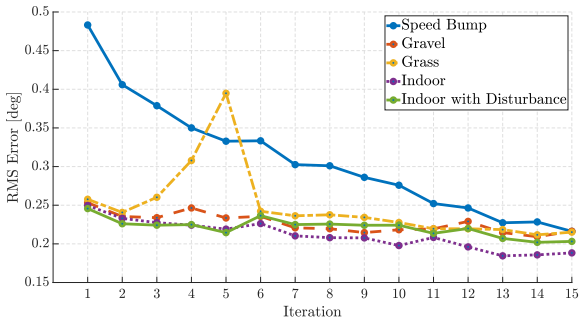
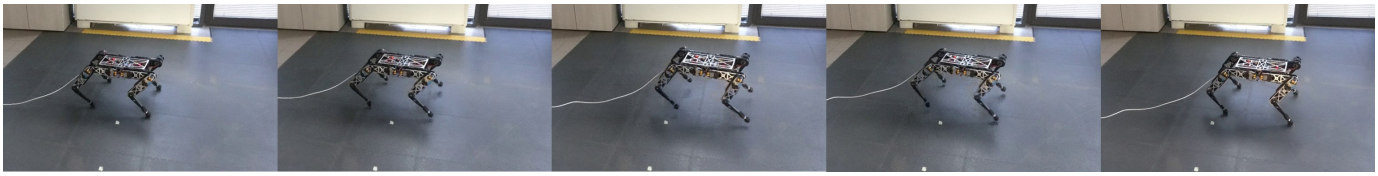


Figure 6: RMS comparison among all the five experiments.

iterations. Results show that, despite the different conditions, the ILC controller enables to improve the tracking performance reducing the RMS error. The indoor experiment (violet dot-dashed line in Fig. 6) demonstrates the most effective reduction in RMS error since the terrain conditions are the easiest. The same experiment is repeated adding a weight of $1\,\mathrm{kg}$ during the learning, specifically at 6-th iteration. This is the indoor experiment with disturbance (green line). The comparison of the two results in Fig. 6 shows identical performance in the first 5 iterations, and an offset starting from the 6-th iteration, which is gradually reduced by the learning algorithm. The yellow dashed line, i.e., grass terrain, shows that the error increases until the 5th iteration. This is caused by the unevenness of the terrain where the robot operates which hinders the robot motion as clearly visible from the video extension. Conversely, on the gravel and grass terrain, the robot navigates with no evident challenges despite the terrain roughness. This observation is further substantiated by the decreasing trend of the orange dashed line in Fig. 6. Finally, the blue line represents the speed bump experiment, which starts with a high RMS error due to the unmodeled terrain feature. However, the controller quickly adapts, reducing the RMS error over iterations and achieving performance comparable to the other experiments. These tests show the robustness of the proposed pipeline, which enables the robot to traverse different



(a) Pronking on indoor environment.

(b) Pronking with added weight on iteration 6.

(c) Pronking on grass.

(d) Pronking on terrain with gravel and grass.

$t++$

Figure 7: Photo-sequence of the Otto experiments. Each test presents a different terrain or unmodeled external disturbance for a single interaction.

environments with the same planned trajectory. Please refer to the video extension too.

Figure 7 shows Otto executing the planned trajectory under four test conditions.

B. Experimental Validations: Step

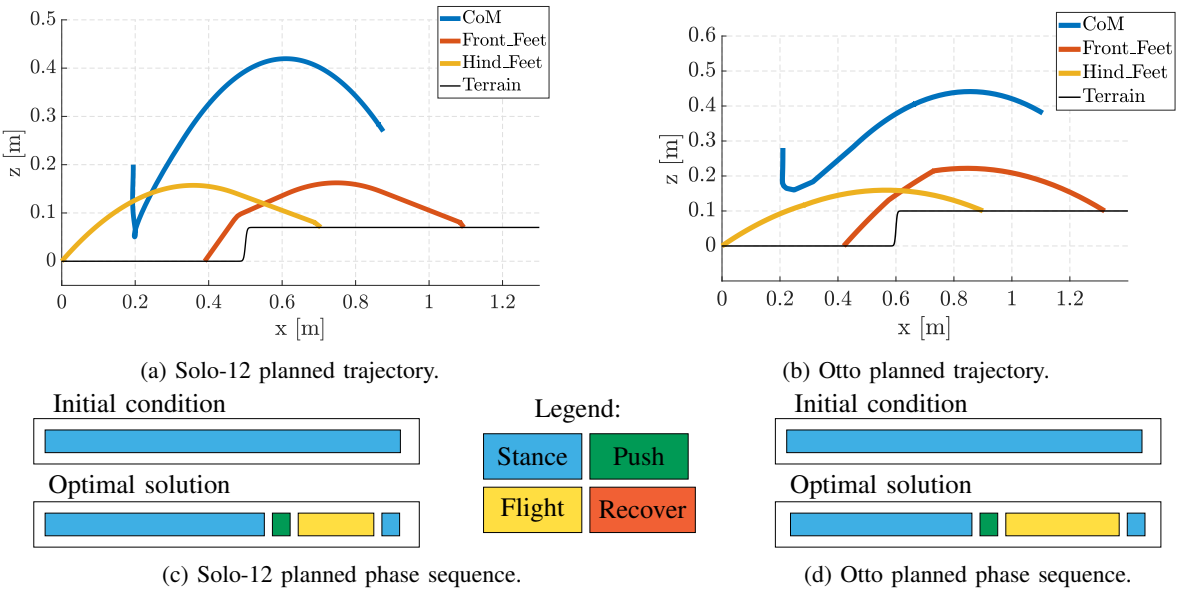
In this section, we test the effectiveness of the proposed method by climbing a step, Specifically, we used the Solo-12 robot to climb a $7\,\mathrm{cm}$ step and the Otto robot to climb a $10\,\mathrm{cm}$ step. To accomplish this task, it is necessary to include the terrain shape as a constraint in the optimization problem as described in Sec. II-F. It is worth mentioning that a step can be climbed by walking or by jumping, and that the specific motion is decided by the planner.

The duration of the task is set as $T = 2\,\mathrm{s}$, while the optimization point number is set as $N = 160$. The initial guess for the phase schedule is set as a single stance phase with duration T. Thus, the robot initial trajectory is stationary. We set $N_\mathrm{T} = 7$, meaning that the robot can accomplish the task using a maximum of two steps.

The results of the planner for both robots are shown in Fig. 8. The CoM and feet trajectories highlight that the planner selects for both robots a jumping motion rather than a climbing one. Indeed, in Fig. 8 there are only two contact points: one at the beginning, and one at the end.

Figures 8c and 8d show the planned phase sequence, highlighting the relative duration of different phases: stance, push, flight, and recover. Most of the task duration is a single stance phase in which the robot crouches (lower its CoM) to prepare for the jump. Then, the robot goes through a brief push phase and a longer flight phase as it moves over the step. Finally, it enters into another stance phase, touching the ground with all four legs simultaneously over the step. No recover phase is present.

Once the trajectory is planned, the experiment is performed with both robots using the ILC controller. In the first iteration, ILC contribution is null, thus only the PD feedback control is active (blue block in Fig. 3). Figure 9 displays the discrepancy between the desired trajectory and the measured joint trajectory during the first iteration for the front and hind legs of the left side of the Solo-12 robot (black dashed line and yellow line, respectively). It is worth noting that, at $\sim 1.6\,\mathrm{s}$, the knee joint error is not negligible, and indeed, the robot fails to complete the task. This highlights the need for ILC to improve trajectory tracking and therefore jump performance (complete scheme in Fig. 3). Measured joint positions in the final iteration (Fig. 9, orange line) demonstrate the advantages of implementing ILC. After ten trials, the measured joint positions align with their references and enable the quadruped to succesfully jump over the step. Otto joint trajectories are



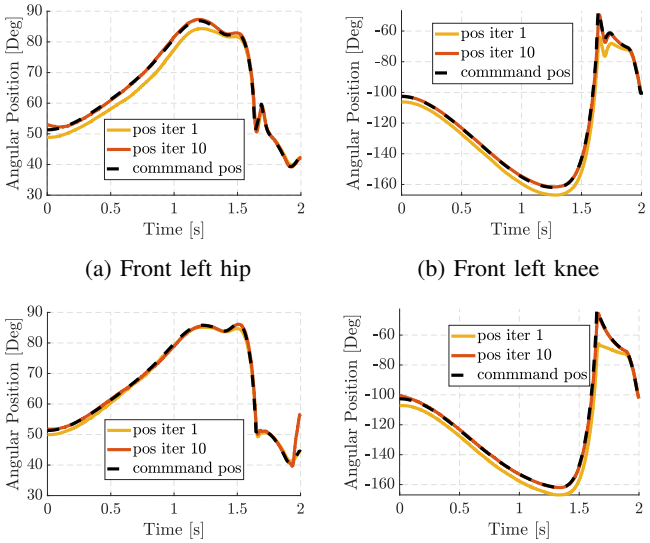
(a) Solo-12 planned trajectory.

(b) Otto planned trajectory.

(c) Solo-12 planned phase sequence.

(d) Otto planned phase sequence.

Figure 8: CoM and feet trajectories resulting from the TO problem for a climbing a step task. (a) Solo-12 climbing a 7 cm step; (b) Otto climbing a 10 cm step.



(a) Front left hip

(b) Front left knee

(c) Hind left hip

(d) Hind left knee

Figure 9: Front and hind left hip and knee joint trajectories: commanded trajectory, iteration 1, and iteration 10.

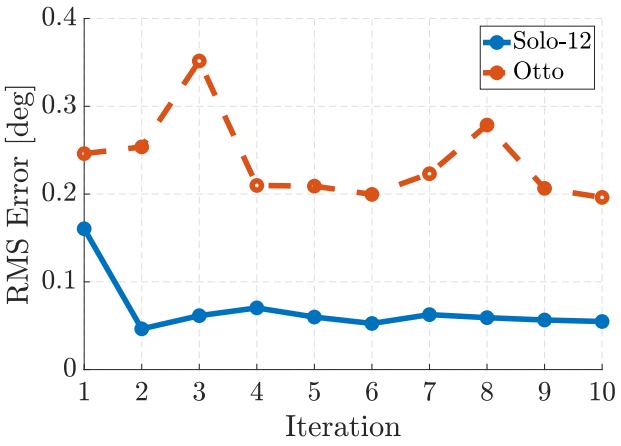


Figure 10: RMS error evolution across multiple iterations during the step jump test.

analogous, and they are not reported here for the sake of space.

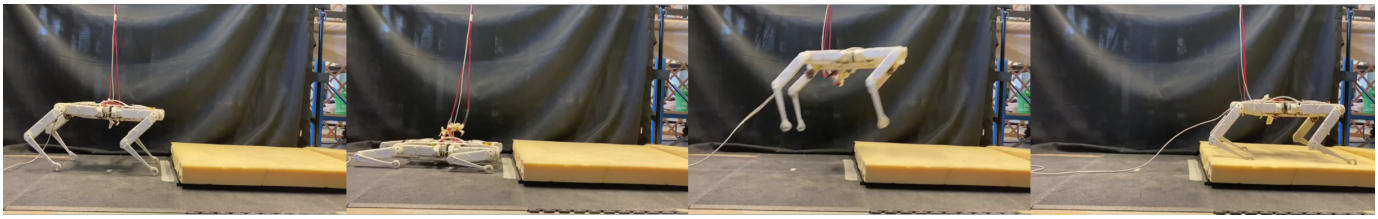
Figure 10 shows the evolution of the RMS error over iteration for both systems: Solo-12 and Otto. Results indicate a decreasing trend in the joint tracking RMS error, signifying the improvement in the robots task performance over time thanks to the application of ILC. The stabilization of the error after 10 iterations suggests that the learning process has plateaued, and further repetitions may not yield significant improvements. At the end of the learning process, both robots were able to jump over the step.

Figure 11 shows a photo sequence of Solo-12 and Otto successfully performing the jump at the last iteration. A more comprehensive and detailed illustration of the completed task can be found in the attached video.

V. Conclusions

The goal of this study is to generate a gait sequence using the TO formulation, which depends on the task and terrain, and deploy the planned trajectories on a quadrupedal robot.

All the experimental validations have yielded satisfactory results in terms of performance. The robots were capable of jumping on steps and walking on different terrains. Furthermore, the robot performed these tasks without specifying the gait in advance. These tasks are sufficient to demonstrate the framework's capabilities, particularly in discovering different gaits depending on the shape of the terrain. In the planning stage, we perform a single large-scale offline optimization,



(a) Snapshots of the 7 cm jump experiment using Solo-12.

(b) Snapshots of the 10 cm jump experiment using Otto.

$t++$

Figure 11: Snapshots from the experiments on steep terrain using Solo-12 (11a) and Otto (11b).

which takes approximately 10 seconds to compute a full trajectory. This longer computation time allows the solver to converge to an optimal solution, resulting in high-quality trajectories that can be executed in real-time without further optimization during deployment. Moreover, we demonstrated that the algorithm is capable of operating with a variety of quadrupedal robots, as evidenced in Sec. IV. About ILC, we demonstrated the convergence of the output over the iteration domain for switched systems. The experiments demonstrated the advantages of ILC in terms of trajectory tracking and robustness. Furthermore, we demonstrated that implementing ILC in the actuated joint space leads the robot to follow a desired trajectory in the Cartesian space.

Our approach differs from online MPC methods, which typically solve a sequence of receding-horizon problems with a limited number of solver iterations, which may not always reach full convergence, but are sufficient for online control. Therefore, the current method is not intended for real-time applications, but rather as a tool to generate high-quality trajectories that can be executed in real-time. Future work will focus on transitioning to an online MPC framework, where the bottleneck of the current method lies in the number of shooting nodes and the number of optimization variables per node.

Appendix A
Proof of Theorem 1

The proof follows three steps. The focus of the first step is demonstrating the output error convergence in the iteration domain for $t \in [0,\, t_1]$, where mode 1 is active. Subsequently, the second step demonstrates the output error convergence in the iteration domain, when mode 2 is active. Here, the state error is affected by an additional term, which highlights the error at the switching instant, i.e., $t = t_1$. Step three extends all the reasoning to each active mode for the whole time interval $t \in [0,\, T]$.

Proof. From Assumption 4, we have

$$\begin{aligned}\delta\dot{\boldsymbol{\zeta}}_k(t) =& \dot{\boldsymbol{\zeta}}_{\mathrm{d}}(t) - \dot{\boldsymbol{\zeta}}_k(t)\\ =& \boldsymbol{f}_i\left(\boldsymbol{\zeta}_{\mathrm{d}}(t)\right) - \boldsymbol{f}_i\left(\boldsymbol{\zeta}_k(t)\right) + \boldsymbol{g}_i\left(\boldsymbol{\zeta}_{\mathrm{d}}(t)\right)\boldsymbol{u}_{\mathrm{d}}(t)\\ & - \boldsymbol{g}_i\left(\boldsymbol{\zeta}_k(t)\right)\boldsymbol{u}_k(t)\\ =& \boldsymbol{f}_i\left(\boldsymbol{\zeta}_{\mathrm{d}}(t)\right) - \boldsymbol{f}_i\left(\boldsymbol{\zeta}_k(t)\right) + \left(\boldsymbol{g}_i\left(\boldsymbol{\zeta}_{\mathrm{d}}(t)\right)\right.\\ & \left. -\boldsymbol{g}_i\left(\boldsymbol{\zeta}_k(t)\right)\right)\boldsymbol{u}_{\mathrm{d}}(t) + \boldsymbol{g}_i\left(\boldsymbol{\zeta}_k(t)\right)\delta\boldsymbol{u}_k(t)\end{aligned} \tag{18}$$

where $\delta\boldsymbol{\zeta}_k(t) = \boldsymbol{\zeta}_{\mathrm{d}}(t) - \boldsymbol{\zeta}_k(t)$ is the state error. From (15) and Assumption 5, we can write the control input error for the $k+1$ as

$$\begin{aligned}&\delta\boldsymbol{u}_{k+1}(t) = \boldsymbol{u}_{\mathrm{d}}(t) - \boldsymbol{u}_{k+1}(t)\\ =&\boldsymbol{u}_{\mathrm{d}}(t) - \boldsymbol{u}_k(t) - \boldsymbol{\Gamma}_i\sum_{j=0}^{r}\gamma_j\left(\boldsymbol{y}_{\mathrm{d}}^{(j)}(t) - \boldsymbol{y}_k^{(j)}(t)\right)\\ =&\delta\boldsymbol{u}_k(t) - \underbrace{\boldsymbol{\Gamma}_i\sum_{j=0}^{r}\gamma_j\left(L_{f_i}^{(j)}\boldsymbol{h}_{i,\,\mathrm{d}} - L_{f_i}^{(j)}\boldsymbol{h}_{i,\,k}\right)}_{\lambda_i(\boldsymbol{\zeta}_k,\,\boldsymbol{\zeta}_{\mathrm{d}})}\\ &+ \boldsymbol{\Gamma}_i L_{g_i}L_{f_i}^{(r-1)}\boldsymbol{h}_{i,\,\mathrm{d}}\boldsymbol{u}_{\mathrm{d}} - \boldsymbol{\Gamma}_i L_{g_i}L_{f_i}^{(r-1)}\boldsymbol{h}_{i,\,k}\boldsymbol{u}_k\\ =&\left(I - \boldsymbol{\Gamma}_i L_{g_i}L_{f_i}^{(r-1)}\boldsymbol{h}_{i,\,k}\right)\delta\boldsymbol{u}_k(t) - \boldsymbol{\Gamma}_i\lambda_i\left(\boldsymbol{\zeta}_k,\,\boldsymbol{\zeta}_{\mathrm{d}}\right)\\ &+ \boldsymbol{\Gamma}_i\left(L_{g_i}L_{f_i}^{(r-1)}\boldsymbol{h}_{i,\,k} - L_{g_i}L_{f_i}^{(r-1)}\boldsymbol{h}_{i,\,\mathrm{d}}\right)\boldsymbol{u}_{\mathrm{d}}.\end{aligned} \tag{19}$$

Let be $\eta \in \mathbb{R}$ such that $\left\|L_{g_i}L_{f_i}^{(r-1)}\boldsymbol{h}_{i,\,k} - L_{g_i}L_{f_i}^{(r-1)}\boldsymbol{h}_{i,\,\mathrm{d}}\right\| \leq \eta\left\|\delta\boldsymbol{\zeta}_k\right\|$. Let $\lambda_0 \in \mathbb{R}$ be a constant value such that

$$\begin{aligned}\left\|\lambda_i\left(\boldsymbol{\zeta}_k, \boldsymbol{\zeta}_{\mathrm{d}}\right)\right\| &= \left\|\sum_{j=0}^{r}\gamma_j\left(L_{f_i}^{(j)}\boldsymbol{h}_{i,\,\mathrm{d}} - L_{f_i}^{(j)}\boldsymbol{h}_{i,\,k}\right)\right\|\\ &\leq \sum_{j=0}^{r}\left\|L_{f_i}^{(j)}\boldsymbol{h}_{i,\,\mathrm{d}} - L_{f_i}^{(j)}\boldsymbol{h}_{i,\,k}\right\|\left\|\gamma_j\right\|\\ &\leq \sum_{j=0}^{r}\left\|\gamma_j\overline{\lambda}_{i,\,k}\right\|\left\|\delta\boldsymbol{\zeta}_k\right\| \leq \sum_{j=0}^{r}\lambda_0\left\|\delta\boldsymbol{\zeta}_k\right\|,\end{aligned}$$

where $\overline{\lambda}_{i,\,k} \in \mathbb{R}$ are constant values such that $\left\| L_{f_i}^{(j)} \boldsymbol{h}_{i,\,\mathrm{d}} - L_{f_i}^{(j)} \boldsymbol{h}_{i,\,k} \right\| \leq \overline{\lambda}_{i,\,k} \left\| \delta\boldsymbol{\zeta}_k \right\|, \forall j = 0, \ldots, r$ (i.e., Lipschitz). Applying bounds and the Lipschitz requirements to the norms on both sides of (19), we obtain

$$\begin{aligned} \left\|\delta\boldsymbol{u}_{k+1}(t)\right\| \leq & \left\| I - \boldsymbol{\Gamma}_i L_{g_i} L_{f_i}^{(r-1)} \boldsymbol{h}_{i,\,k} \right\| \left\|\delta\boldsymbol{u}_k(t)\right\| \\ & + \left\|\Gamma_i\right\| \left\|\lambda_i\left(\boldsymbol{\zeta}_k,\, \boldsymbol{\zeta}_{\mathrm{d}}\right)\right\| \\ & + \left\|\boldsymbol{\Gamma}_i\right\| \left\| \left(L_{g_i} L_{f_i}^{(r-1)} \boldsymbol{h}_{i,\,k} - L_{g_i} L_{f_i}^{(r-1)} \boldsymbol{h}_{i,\,\mathrm{d}} \right) \right\| \left\|\boldsymbol{u}_{\mathrm{d}}(t)\right\| \\ \leq & \rho \left\|\delta\boldsymbol{u}_k(t)\right\| + b_\Gamma \left(\eta \left\|\boldsymbol{u}_{\mathrm{d}}(t)\right\| + \lambda_0\right) \left\|\delta\boldsymbol{\zeta}_k(t)\right\| \end{aligned} \tag{20}$$

where b_Γ is norm bound for $\boldsymbol{\Gamma}_i$. We define

$$k_1 \triangleq \sup_{t\in[0,T]} b_\Gamma \left(\eta \left\|\boldsymbol{u}_{\mathrm{d}}(t)\right\| + \lambda_0\right),$$

thus (20) can be described as

$$\left\|\delta\boldsymbol{u}_{k+1}(t)\right\| \leq \rho \left\|\delta\boldsymbol{u}_k(t)\right\| + k_1 \left\|\delta\boldsymbol{\zeta}_k(t)\right\|. \tag{21}$$

Step 1: When $t \in [0, t_1]$ In this case, the subsystem 1 is active. Writing the integral expression for $\boldsymbol{\zeta}_k(t)$ and taking norms, considering Assumption 3 and integrating (18), we obtain

$$\begin{aligned} \left\|\delta\boldsymbol{\zeta}_k(t)\right\| = & \left\|\boldsymbol{\zeta}_{\mathrm{d}}(0) - \boldsymbol{\zeta}_k(0) + \int_0^t \left(\left(\boldsymbol{f}_{1,\mathrm{d}} + \boldsymbol{g}_{1,\mathrm{d}}\boldsymbol{u}_{\mathrm{d}}\right) \right.\right. \\ & \left.\left. - \left(\boldsymbol{f}_{1,k} + \boldsymbol{g}_{1,k}\boldsymbol{u}_k\right) \right) \mathrm{d}\tau \right\| \\ \leq & \int_0^t \left(\left\|\boldsymbol{f}_{1,d} - \boldsymbol{f}_{1,k}\right\| + \left\|\boldsymbol{g}_{1,d} - \boldsymbol{g}_{1,k}\right\| \left\|\boldsymbol{u}_{\mathrm{d}}\right\| \right. \\ & \left. + \left\|\boldsymbol{g}_{1,k}\right\| \left\|\delta\boldsymbol{u}_k(t)\right\| \right) \mathrm{d}\tau \\ \leq & \int_0^t \left(\left(k_{\mathrm{f}} + k_{\mathrm{g}} b_{\mathrm{ud}}\right) \left\|\delta\boldsymbol{\zeta}_k(t)\right\| + b_{\mathrm{B}} \left\|\delta\boldsymbol{u}_k(t)\right\| \right) \mathrm{d}\tau, \end{aligned} \tag{22}$$

where $b_{\mathrm{B}}, b_{\mathrm{ud}}$ are the norm bounds on $\boldsymbol{g}_i\left(\boldsymbol{\zeta}_k(t)\right)$ and $\boldsymbol{u}_{\mathrm{d}}$, respectively. We define $k_3 = k_{\mathrm{f}} + k_{\mathrm{b}} b_{\mathrm{ud}}$, and using the Bellman-Gronwall lemma we have

$$\left\|\delta\boldsymbol{\zeta}_k(t)\right\| \leq \int_0^t \mathrm{e}^{k_3(t-\tau)} b_{\mathrm{B}} \left\|\delta\boldsymbol{u}_k(\tau)\right\| \mathrm{d}\tau, t \in [0, t_1]. \tag{23}$$

Combining (21) and (23) yields

$$\left\|\delta\boldsymbol{u}_{k+1}(t)\right\| \leq \rho \left\|\delta\boldsymbol{u}_k(t)\right\| + k_1 b_{\mathrm{B}} \int_0^t \mathrm{e}^{k_3(t-\tau)} \left\|\delta\boldsymbol{u}_k(\tau)\right\| \mathrm{d}\tau. \tag{24}$$

Multiplying (24) by $\mathrm{e}^{-\lambda t}$, define $\kappa \triangleq \max\left\{k_1 b_{\mathrm{B}}, k_3\right\}$ and assuming $\lambda > \kappa$, we have

$$\begin{aligned} \mathrm{e}^{-\lambda t} \left\|\delta\boldsymbol{u}_{k+1}(t)\right\| \leq & \rho \mathrm{e}^{-\lambda t} \left\|\delta\boldsymbol{u}_k(t)\right\| + \kappa \int_0^t \mathrm{e}^{-\lambda\tau} \left\|\delta\boldsymbol{u}_k(\tau)\right\| \\ & \times \mathrm{e}^{(\kappa-\lambda)(t-\tau)} \mathrm{d}\tau. \end{aligned} \tag{25}$$

Note that for a constant $\|\kappa\|_\lambda = \kappa$, we obtain [3]

$$\begin{aligned} \left\|\delta\boldsymbol{u}_{k+1}(t)\right\|_\lambda \leq & \left[\rho + \frac{\kappa}{\lambda - \kappa}\left(1 - \mathrm{e}^{(\kappa-\lambda)t_1}\right)\right] \left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda \\ = & \bar{\rho}_1 \left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda, \end{aligned} \tag{26}$$

where $\bar{\rho}_1 = \rho + \left(\kappa/(\lambda-\kappa)\right)\left(1 - \mathrm{e}^{(\kappa-\lambda)t_1}\right)$. Since $\rho < 1$, it is possible to find a $\lambda > \kappa$ that makes $\bar{\rho}_1 < 1$, which leads to $\lim_{k\to\infty} \left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda = 0$. Using (23), and similar manipulations we have

$$\begin{aligned} \left\|\delta\boldsymbol{\zeta}_k(t)\right\|_\lambda \leq & b_{\mathrm{B}} \int_0^t \mathrm{e}^{(k_3-\lambda)(t-\tau)} \left\|\delta\boldsymbol{u}_k(\tau)\right\|_\lambda \mathrm{d}\tau \\ \leq & \frac{b_{\mathrm{B}}}{\lambda - k_3}\left(1 - \mathrm{e}^{(k_3-\lambda)t_1}\right) \left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda. \end{aligned}$$

Hence, one has

$$\lim_{k\to\infty} \left\|\delta\boldsymbol{\zeta}_k(t)\right\|_\lambda = 0. \tag{27}$$

We use the fact that $\boldsymbol{h}_1\left(\boldsymbol{\zeta}_k(t)\right)$ is Lipschitz in $\boldsymbol{\zeta}_k(t)$, we also can obtain $\lim_{k\to\infty}\left[\boldsymbol{y}_{\mathrm{d}}(t) - \boldsymbol{y}_k(t)\right] = 0$, for $t \in [0, t_1]$.

Step 2: When $t \in [t_1, t_2]$. The subsystem 2 is operating in this instance. Recalling (18), the state error can be expressed as

$$\begin{aligned} \delta\boldsymbol{\zeta}_k(t) = & \delta\boldsymbol{\zeta}_k\left(t_1\right) + \int_{t_1}^t \left(\left(\boldsymbol{f}_{2,\mathrm{d}} + \boldsymbol{g}_{2,\mathrm{d}}\boldsymbol{u}_{\mathrm{d}}\right) \right. \\ & \left. - \left(\boldsymbol{f}_{2,k} + \boldsymbol{g}_{2,k}\boldsymbol{u}_k\right) \right) \mathrm{d}\tau. \end{aligned} \tag{28}$$

Taking (28) and applying the norm of both sides of the equation, we obtain

$$\begin{aligned} \left\|\delta\boldsymbol{\zeta}_k(t)\right\| \leq & \left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\| + \int_{t_1}^t \left(\left(k_f + k_{\mathrm{b}} b_{\mathrm{ud}}\right) \left\|\delta\boldsymbol{\zeta}_k(t)\right\| \right. \\ & \left. + b_{\mathrm{B}} \left\|\delta\boldsymbol{u}_k(t)\right\| \right) \mathrm{d}\tau. \end{aligned} \tag{29}$$

Applying Bellman-Gronwall lemma to (29) produces

$$\left\|\delta\boldsymbol{\zeta}_k(t)\right\| \leq \left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\| + \int_{t_1}^t \mathrm{e}^{k_3(t-\tau)} b_{\mathrm{B}} \left\|\delta\boldsymbol{u}_k(\tau)\right\| \mathrm{d}\tau. \tag{30}$$

Combining (21) and (30) yields

$$\begin{aligned} \left\|\delta\boldsymbol{u}_{k+1}(t)\right\| \leq & \rho \left\|\delta\boldsymbol{u}_k(t)\right\| + k_1 b_{\mathrm{B}} \int_{t_1}^t \mathrm{e}^{k_3(t-\tau)} \left\|\delta\boldsymbol{u}_k(\tau)\right\| \mathrm{d}\tau \\ & + k_1 \left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|. \end{aligned} \tag{31}$$

Multiplying (31) by $\mathrm{e}^{-\lambda t}$, we have

$$\begin{aligned} \left\|\delta\boldsymbol{u}_{k+1}(t)\right\|_\lambda \leq & \left[\rho + \frac{\kappa}{\lambda - \kappa}\left(1 - \mathrm{e}^{(\kappa-\lambda)t_2}\right)\right] \left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda \\ & + k_1 \left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|_\lambda \\ = & \bar{\rho}_2 \left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda + k_1 \left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|_\lambda \end{aligned} \tag{32}$$

where $\bar{\rho}_2 = \rho + (\kappa/(\lambda-\kappa))\left(1 - \mathrm{e}^{(\kappa-\lambda)t_2}\right)$. We can also obtain a $\lambda > \kappa$, which makes $\bar{\rho}_2 < 1$, because $\rho < 1$.

Define $\delta\tilde{\boldsymbol{\zeta}}_k\left(t_1\right) \triangleq \sup\left\{\left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|_\lambda, \left\|\delta\boldsymbol{\zeta}_{k+1}\left(t_1\right)\right\|_\lambda, \ldots\right\}$. Hence, $\delta\tilde{\boldsymbol{\zeta}}_k\left(t_1\right) \geq \delta\tilde{\boldsymbol{\zeta}}_{k+1}\left(t_1\right) \geq 0, \delta\tilde{\boldsymbol{\zeta}}_k\left(t_1\right) \geq \left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|_\lambda$.

[3] In this paper we use the λ-norm, which is defined by $\|f(\cdot)\|_\lambda = \sup_{t\in[0,T]}\left\{e^{-\lambda t}\left\|f(t)\right\|\right\}$, $\lambda > 0$, with $f : [0, T] \to \mathbb{R}^n$.

From (32), if k is even, then

$$\begin{aligned}\left\|\delta\boldsymbol{u}_{k+1}(t)\right\|_\lambda \leq & \bar{\rho}_2\left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda + k_1\left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|_\lambda \\ \leq & \bar{\rho}_2^k\left\|\delta\boldsymbol{u}_1(t)\right\|_\lambda + k_1\bar{\rho}_2^{k-1}\delta\tilde{\boldsymbol{\zeta}}_1\left(t_1\right) \\ & + k_1\bar{\rho}_2^{k-2}\delta\tilde{\boldsymbol{\zeta}}_2\left(t_1\right) + \cdots + k_1\delta\tilde{\boldsymbol{\zeta}}_k\left(t_1\right) \\ = & \bar{\rho}_2^k\left\|\delta\boldsymbol{u}_1(t)\right\|_\lambda + k_1\bar{\rho}_2^{k-1}\delta\tilde{\boldsymbol{\zeta}}_1\left(t_1\right) \\ & + \cdots + k_1\bar{\rho}_2^{(k/2)}\delta\tilde{\boldsymbol{\zeta}}_{(k/2)}\left(t_1\right) + k_1\bar{\rho}_2^{(k/2)-1} \\ & \times\delta\tilde{\boldsymbol{\zeta}}_{(k/2)+1}\left(t_1\right) + \cdots + k_1\delta\tilde{\boldsymbol{\zeta}}_k\left(t_1\right) \\ \leq & \bar{\rho}_2^{(k/2)}\left(\left\|\delta\boldsymbol{u}_1(t)\right\|_\lambda + \frac{k}{2}k_1\delta\tilde{\boldsymbol{\zeta}}_1\left(t_1\right)\right) \\ & + k_1\delta\tilde{\boldsymbol{\zeta}}_{(k/2)+1}\left(t_1\right) \\ & \times\left(\bar{\rho}_2^{(k/2)-1} + \bar{\rho}_2^{(k/2)-2} + \cdots + 1\right) \\ = & \bar{\rho}_2^{(k/2)}\left(\left\|\delta\boldsymbol{u}_1(t)\right\|_\lambda + \frac{k}{2}k_1\delta\tilde{\boldsymbol{\zeta}}_1\left(t_1\right)\right) \\ & + k_1\delta\tilde{\boldsymbol{\zeta}}_{(k/2)+1}\left(t_1\right)\frac{1-\bar{\rho}_2^{(k/2)-1}}{1-\bar{\rho}_2}.\end{aligned} \tag{33}$$

Taking (33), we evaluate both sides for $k\to\infty$, then we have

$$\begin{aligned}\lim_{k\to\infty}\left\|\delta\boldsymbol{u}_{k+1}(t)\right\|_\lambda \leq & \lim_{k\to\infty}\bar{\rho}_2^{(k/2)}\left(\left\|\delta\boldsymbol{u}_1(t)\right\|_\lambda + \frac{k}{2}k_1\delta\tilde{\boldsymbol{\zeta}}_1\left(t_1\right)\right) \\ & + k_1\lim_{k\to\infty}\left(\frac{1-\bar{\rho}_2^{(k/2)-1}}{1-\bar{\rho}_2}\delta\tilde{\boldsymbol{\zeta}}_{(k/2)+1}\left(t_1\right)\right).\end{aligned} \tag{34}$$

Recall from step 1 that $\lim_{k\to\infty}\left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|_\lambda = 0$, we can state that $\lim_{k\to\infty}\delta\tilde{\boldsymbol{\zeta}}_k\left(t_1\right) = 0$. Moreover, (34) implies

$$\lim_{k\to\infty}\left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda = 0.$$

Similarly, we can derive that $\lim_{k\to\infty}\left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda = 0$, if k is odd.

Multiplying (30) by $\mathrm{e}^{-\lambda t}$, we obtain

$$\begin{aligned}\left\|\delta\boldsymbol{\zeta}_k(t)\right\|_\lambda \leq & \left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|_\lambda \\ & + \frac{b_{\mathrm{B}}}{\lambda-k_3}\left(1-\mathrm{e}^{(k_3-\lambda)t_2}\right)\left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda.\end{aligned} \tag{35}$$

Recalling that $\lim_{k\to\infty}\left\|\delta\boldsymbol{\zeta}_k\left(t_1\right)\right\|_\lambda = 0$, it takes us to consider that $\lim_{k\to\infty}\left\|\delta\boldsymbol{u}_k(t)\right\|_\lambda = 0$. Thus, we can obtain $\lim_{k\to\infty}\left\|\delta\boldsymbol{\zeta}_k(t)\right\|_\lambda = 0$.

We use the fact that $\boldsymbol{h}_2\left(\boldsymbol{\zeta}_k(t)\right)$ is Lipschitz in $\boldsymbol{\zeta}_k(t)$ and this leads to $\lim_{k\to\infty}\left[\boldsymbol{y}_{\mathrm{d}}(t)-\boldsymbol{y}_k(t)\right] = 0$ for $t\in[t_1,t_2]$.

Step 3: Following these steps, we can also obtain $\lim_{k\to\infty}\left\|e_k(t)\right\|_\lambda = 0$ for $t\in[t_2,t_3], t\in[t_3,t_4],\ldots,[t_{m-1},T]$. Thus, we get $\lim_{k\to\infty}\left[\boldsymbol{y}_{\mathrm{d}}(t)-\boldsymbol{y}_k(t)\right] = 0$ for the entire time interval $t\in[0,T]$. This completes the proof. □

REFERENCES

[1] R. M. Alexander, "The gaits of bipedal and quadrupedal animals," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 49–59, 1984.

[2] D. F. Hoyt and C. R. Taylor, "Gait and the energetics of locomotion in horses," *Nature*, vol. 292, no. 5820, pp. 239–240, 1981.

[3] V. S. Medeiros, E. Jelavic, M. Bjelonic, R. Siegwart, M. A. Meggiolaro, and M. Hutter, "Trajectory optimization for wheeled-legged quadrupedal robots driving in challenging terrain," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4172–4179, 2020.

[4] H. Kolvenbach, D. Wisth, R. Buchanan, G. Valsecchi, R. Grandia, M. Fallon, and M. Hutter, "Towards autonomous inspection of concrete deterioration in sewers with legged robots," *Journal of field robotics*, vol. 37, no. 8, pp. 1314–1327, 2020.

[5] M. Eich, F. Grimminger, and F. Kirchner, "A versatile stair-climbing robot for search and rescue applications," in *2008 IEEE international workshop on safety, security and rescue robotics*. IEEE, 2008, pp. 35–40.

[6] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[7] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[8] P. Sutyasadi and M. Parnichun, "Trotting control of a quadruped robot using pid-ilc," in *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2015, pp. 004 400–004 405.

[9] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[10] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 93–100.

[11] F. Farshidian, M. Kamgarpour, D. Pardo, and J. Buchli, "Sequential linear quadratic optimal control for nonlinear switched systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1463–1469, 2017.

[12] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1502–1509, 2017.

[13] M. R. Turski, J. Norby, and A. M. Johnson, "Staged contact optimization: Combining contact-implicit and multi-phase hybrid trajectory optimization," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2376–2383.

[14] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3316–3323, 2018.

[15] X. Xu and P. J. Antsaklis, "Optimal control of switched systems via nonlinear optimization based on direct differentiations of value functions," *International Journal of Control*, vol. 75, no. 16-17, pp. 1406–1426, 2002.

[16] ——, "Optimal control of switched systems based on parameterization of the switching instants," *IEEE transactions on automatic control*, vol. 49, no. 1, pp. 2–16, 2004.

[17] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2017.

[18] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[19] R. Batke, F. Yu, J. Dao, J. Hurst, R. L. Hatton, A. Fern, and K. Green, "Optimizing bipedal maneuvers of single rigid-body models for reinforcement learning," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. IEEE, 2022, pp. 714–721.

[20] F. Roscia, M. Focchi, A. Del Prete, D. G. Caldwell, and C. Semini, "Reactive landing controller for quadruped robots," *IEEE Robotics and Automation Letters*, 2023.

[21] R. Blickhan, "The spring-mass model for running and hopping," *Journal of biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, 1989.

[22] D. De Benedittis, F. Angelini, and M. Garabini, "Soft bilinear inverted pendulum: A model to enable locomotion with soft contacts," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 2, pp. 1478–1491, 2025.

[23] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[24] Z. Yu, X. Chen, Q. Huang, W. Zhang, L. Meng, W. Zhang, and J. Gao, "Gait planning of omnidirectional walk on inclined ground for biped robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 7, pp. 888–897, 2016.

[25] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, 2023.

[26] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[27] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics: optimization and feedback control*. Springer, 2006, pp. 65–93.

[28] F. Angelini, R. Mengacci, C. Della Santina, M. G. Catalano, M. Garabini, A. Bicchi, and G. Grioli, "Time generalization of trajectories learned on articulated soft robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3493–3500, 2020.

[29] M. Pierallini, F. Stella, F. Angelini, B. Deutschmann, J. Hughes, A. Bicchi, M. Garabini, and C. Della Santina, "A provably stable iterative learning controller for continuum soft robots," *IEEE Robotics and Automation Letters*, 2023.

[30] Y.-S. Wei and X.-D. Li, "Robust higher-order ilc for non-linear discrete-time systems with varying trail lengths and random initial state shifts," *IET Control Theory & Applications*, vol. 11, no. 15, pp. 2440–2447, 2017.

[31] Y. Liu, R.-H. Chi, and Z.-S. Hou, "Neural network state learning based adaptive terminal ilc for tracking iteration-varying target points," *International Journal of Automation and Computing*, vol. 12, no. 3, pp. 266–272, 2015.

[32] J. Ding, M. A. van Löben Sels, F. Angelini, J. Kober, and C. Della Santina, "Robust jumping with an articulated soft quadruped via trajectory optimization and iterative learning," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 255–262, 2023.

[33] M. A. Sen, V. Bakircioglu, and M. Kalyoncu, "Inverse kinematic analysis of a quadruped robot," *International journal of scientific & technology research*, vol. 6, no. 9, pp. 285–289, 2017.

[34] H.-S. Ahn, Y. Chen, and K. L. Moore, "Iterative learning control: Brief survey and categorization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1099–1121, 2007.

[35] "Open Dynamic Robot Initiative — open-dynamic-robot-initiative.github.io," https://open-dynamic-robot-initiative.github.io, [Accessed 07-05-2024].

[36] A. Scaldaferri, S. Tolomei, F. Iotti, P. Gambino, M. Pierallini, F. Angelini, and M. Garabini, "Otto - design and control of an 8-dof sea-driven quadrupedal robot," *IEEE Open Journal of the Industrial Electronics Society*, pp. 1–21, 2025.

[37] A. Benzaouia, *Introduction to Switched Systems*. London: Springer London, 2012, pp. 77–94. [Online]. Available: https://doi.org/10.1007/978-1-4471-2900-4_3

[38] X. Xu and P. J. Antsaklis, "Results and perspectives on computational methods for optimal control of switched systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2003, pp. 540–555.

[39] D. Liberzon, "Switched systems," in *Handbook of networked and embedded control systems*. Springer, 2005, pp. 559–574.

[40] P. Gori, M. Pierallini, F. Angelini, and M. Garabini, "Continuous-time constrained linear quadratic regulator for switched linear systems," *Nonlinear Analysis: Hybrid Systems*, vol. 58, p. 101625, 2025.

[41] G. Hicks and W. Ray, "Approximation methods for optimal control synthesis," *The Canadian Journal of Chemical Engineering*, vol. 49, no. 4, pp. 522–528, 1971.

[42] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2536–2542.

[43] M. Pierallini, F. Angelini, R. Mengacci, A. Palleschi, A. Bicchi, and M. Garabini, "A robust iterative learning control for continuous-time nonlinear systems with disturbances," *IEEE Access*, vol. 9, pp. 147 471–147 480, 2021.

[44] X. Bu, Z. Hou, F. Yu, and Z. Fu, "Iterative learning control for a class of non-linear switched systems," *IET Control Theory & Applications*, vol. 7, no. 3, pp. 470–481, 2013.

[45] M. Pierallini, F. Angelini, R. Mengacci, A. Palleschi, A. Bicchi, and M. Garabini, "Iterative learning control for compliant underactuated arms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[46] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[47] T.-J. Jang, C.-H. Choi, and H.-S. Ahn, "Iterative learning control in feedback systems," *Automatica*, vol. 31, no. 2, pp. 243–248, 1995.

[48] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[49] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

Pietro Gori received the B.S. degree in electronic engineering in 2020 and the M.S. degree (cum laude) in automation and robotics engineering in 2023 from the University of Pisa, Pisa, Italy. He is currently pursuing the Smart Industry Ph.D. degree in robotics at the Research Center Enrico Piaggio, Pisa, Italy. His research focuses on the planning of quadrupedal robots' motion and optimal control.

Michele Pierallini received a B.S. degree in biomedical engineering in 2017 and M.S. degree (cum laude) in automation and robotics engineering in 2020 from the University of Pisa, Pisa, Italy, where he is currently working toward the Ph.D. degree in robotics at the Research Center Enrico Piaggio. His current research focuses on the control of soft robotic systems and iterative learning control.

Franco Angelini received the B.S. degree in computer engineering in 2013 and M.S. degree (cum laude) in automation and robotics engineering in 2016 from the University of Pisa, Pisa, Italy. The University of Pisa also granted him a Ph.D. degree (cum laude) in robotics in 2020 for which he was awarded the 2021 SIDRA Award for the best PhD Thesis in the area of Systems and Control Engineering at an Italian University. Franco is currently an Assistant Professor at the University of Pisa. His main research interests are control of soft robotic systems, impedance planning, and robotic environmental monitoring.

Manolo Garabini graduated in Mechanical Engineering and received the Ph.D. in Robotics from the University of Pisa where he is currently employed as Associate Professor. His main research interests include the design, planning, and control of soft adaptive robots. He contributed to the realization of modular Variable Stiffness Actuators, to the design of the humanoid robot WALK-MAN and, to the development of efficient and effective compliance planning algorithms. Currently, he is the Principal Investigator in the THING H2020 EU Research Project, the coordinator of the Dysturbance H2020 Eurobench sub-project and the H2020 EU Research Project Natural Intelligence.