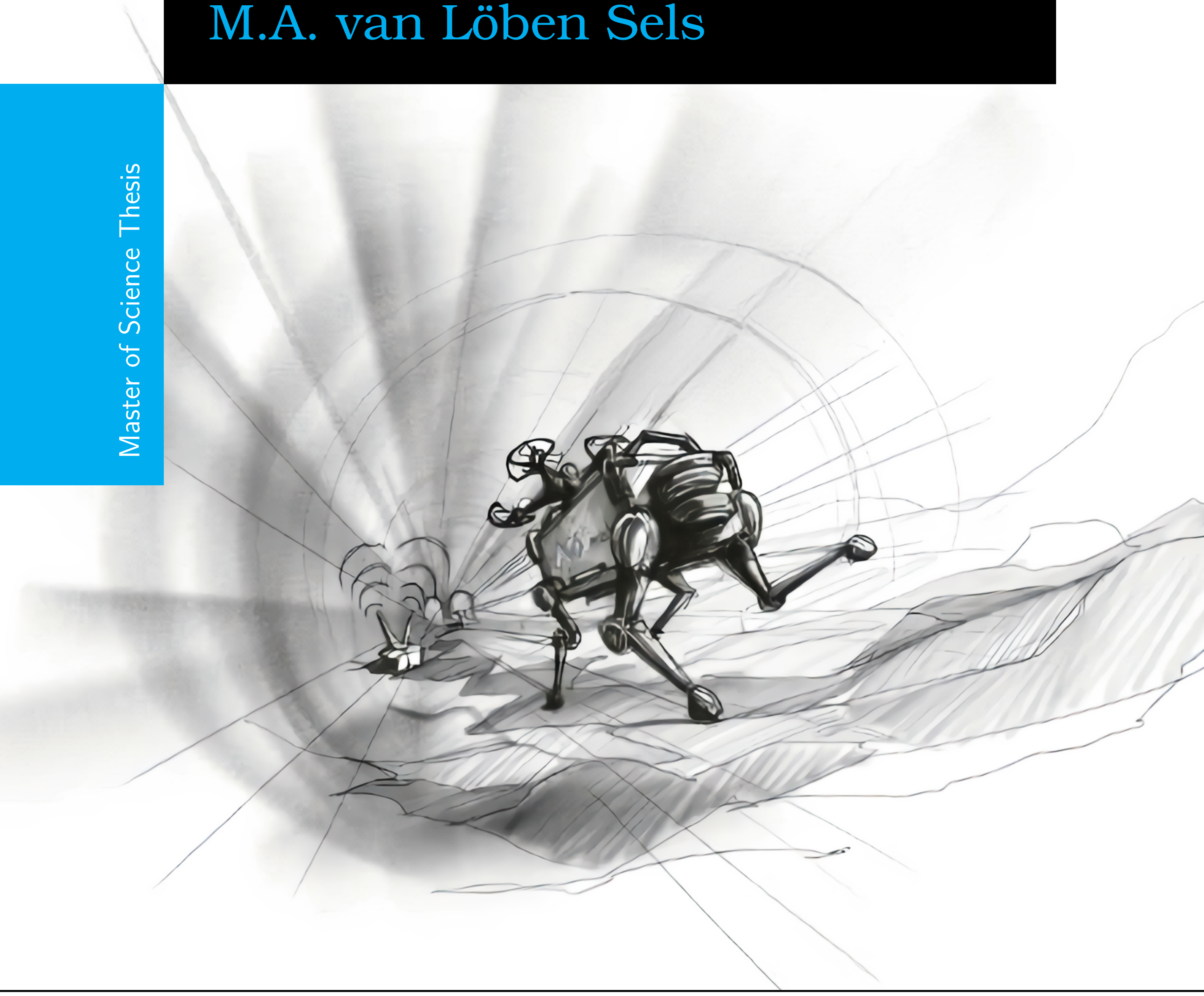
SLIP-based Iterative Learning for Efficient and Compliant Locomotion of Articulated Soft Quadrupeds

M.A. van Löben Sels

Master of Science Thesis



Delft University of Technology

SLIP-based Iterative Learning for Efficient and Compliant Locomotion of Articulated Soft Quadrupeds

Master of Science Thesis

For the degree of Master of Science in Robotics at Delft University of Technology

M.A. van Löben Sels

Student number:	4438132	
Daily supervisors:	dr. C. Della Santina,	TU Delft
	dr. J. Ding,	TU Delft
Thesis committee:	dr. C. Della Santina,	TU Delft, supervisor
	dr. J. Ding,	TU Delft, supervisor
	dr. J. Kober,	TU Delft
	dr. L. Laurenti,	TU Delft

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Cognitive Robotics
Faculty of 3mE
TU Delft
The Netherlands

Preface

This thesis marks the conclusion of my 7.5 years as a student in Delft and is the culmination of my research over the past year. It was a long but enjoyable ride, and I can only say that I am proud of the things I have learned and what I have accomplished, during my thesis and my student time as a whole.

Of course, this would not have been possible without the help of all those involved in this thesis. First and foremost, I would like to thank my supervisors, dr. Cosimo Della Santina and dr. Jiatao Ding for their great supervision and guidance throughout this project. I am very thankful for all the frequent and insightful meetings that we had. Thank you Jiatao for your help with the experiments. I initially did not believe it would be possible to do the experiments in only two weeks. But with your assistance, we made it work. I would also like to thank Vassil Atanassov for his understanding of the robot's monumental pile of (commented) code and quick tips on PyBullet.

There have also been many non-scientific contributions that helped me through the year. Thank you, mom and dad, for all your support over the past years, both emotionally and financially. Thank you Jans for always being there for me and for your endless support. Not only during my thesis but during my whole student time. Thanks to all my friends whom I have spent lunch with the past year, the members of the TUkan gang, my study buddies, and all others. Being in the same boat made the year much more enjoyable.

Mees Alexander van Löben Sels
Delft, 12 February 2023

SLIP-based Iterative Learning for Efficient and Compliant Locomotion of Articulated Soft Quadrupeds

M.A. van Löben Sels

***Abstract*—Effectively controlling and exploiting the natural dynamics of Articulated Soft Robots for energy-efficient motions remains challenging. In literature, the problem is often split in two; in energy-efficient motion planning and structure-preserving control, where the focus is on one, and the other is largely disregarded. This work aims to unify these two using a motion planning and control strategy based on trajectory optimization and functional Iterative Learning Control. Using a reduced-order model, the planner generates an energy-efficient reference trajectory by minimizing the Cost of Transport. The controller then iteratively learns the feedforward control signal such that the full-order system tracks the reference, without altering its stiffness characteristics. We show that our strategy results in energy-efficient tracking of the reference. We also show how functional Iterative Learning Control can be used in a continuous approach to learn a stable forward pronking gait. We give experimental validation of this approach through experiments on the compliant quadruped E-Go. We show that the pronking gait can be learned on hardware in minutes and that it is robust to various types of terrain.**

I. Introduction

In recent years, legged robots have made significant technological advancements. After decades of research, they are now leaving the confinements of research labs and have found varying applications in industries such as industrial inspection and surveillance [1]. However, today's legged robots still lack the efficiency of their biological counterparts, resulting in a continuous operational time of only a few hours [2], [3]. The periodic motion of legged locomotion causes large fluctuations in kinetic and potential energy. High energetic efficiency in humans and animals can largely be accredited to the recuperation of these energy fluctuations through soft muscles and tendons [4]. Instead of creating the entire motion actively, the inherent elasticity of the musculoskeletal system allows humans and animals to temporarily store and release energy such that a part of the walking motion emerges passively [5]. Inspired by the musculoskeletal system of vertebrate animals, soft elastic elements can be introduced to legged robots to exploit the so-called *natural dynamics* of their mechanical system, requiring less energy to be injected actively and increasing the energetic efficiency. Robots with elastic elements concentrated in the joints are called articulated soft robots (ASRs) [6].

However, effectively exploiting and controlling the natural dynamics of ASRs remains a complex problem, particularly for legged locomotion. The locomotion task can generally be split into two problems, motion planning and control. The generation of energy-efficient gaits that exploit elasticity has been studied extensively for both reduced-order [8]–[12] and full-order systems [13], [14], but these works only approach the problem from the motion planning perspective

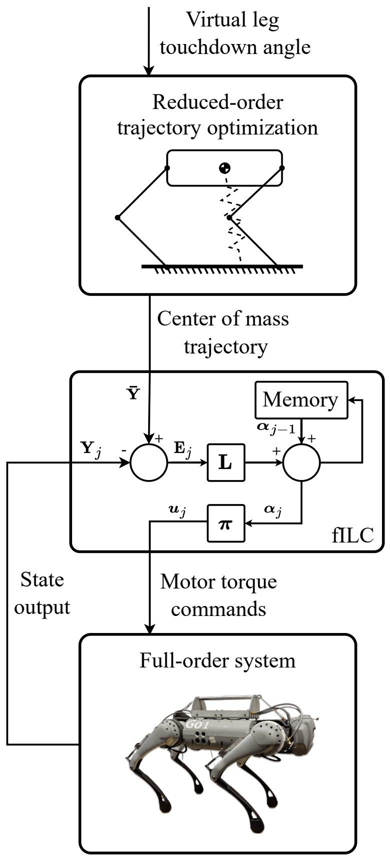


Fig. 1: Overview of the planning and control framework. The trajectory planner generates a center of mass trajectory based on a reduced-order model. The controller learns the required control input to map the reference to the full-order system. The full-order system is modeled after the compliant quadruped E-Go [7].

and do not consider how to execute these gaits. Typically, feedback control methods are used to control elastic systems, but feedback control alters the stiffness of the system with a factor proportional to the feedback gain [15], defeating the purpose of introducing physical compliance in the first place. In this regard, feedforward control actions and methods with minimal reliance on feedback are preferable, so called structure-preserving controllers [6]. Examples of these type of controllers are [16], [17]. Although these methods are effective in preserving the elasticity while achieving accurate tracking, they do not specifically exploit the natural dynamics for more efficient motions, and have found limited application to legged locomotion. Similarly, control methods that claim to exploit the natural dynamics of elastic legged systems still rely on feedback [18]–[21]. As such, works on this subject typically consider only a part of the problem and do not solve the full problem of exploiting natural dynamics for efficient locomotion, namely motion planning and control. To the best of the author's knowledge, the only exception to this is the recent work using normal mode theory to exploit and stabilize modal oscillations for efficient locomotion [22]. Despite yielding promising results, the work lacks an analysis of energy efficiency. Hence, no conclusions can be made regarding the efficiency of the method and if the natural dynamics are indeed exploited.

This work presents a step towards a unifying motion planning and control framework for efficient and compliant locomotion of ASRs, based on offline trajectory optimization and iterative learning control (ILC). The trajectory planner uses a single rigid-body dynamics (SRBD) model, which considers the effect of the external forces and joint torques on the base motion. This reduced-order model increases the problem tractability while still capturing the governing dynamics of the full-order system, allowing the planner to generate energy-efficient gaits. The benefit of using ILC as the controller is twofold. First, the feedforward nature of ILC in the time-domain preserves the physical compliance of the system, while yielding accurate tracking performance through feedback in the iteration-domain. Second, we omit the need of deriving an accurate model to map the reduced-order model behavior to the full-order system. Typically, this mapping is achieved using methods such as operational space control [20], [23], relying on inverse dynamics to calculate the required ground reaction forces and joint torques to track the given reference, and as a consequence, requiring an accurate model. ILC overcomes the model mismatch by learning the required control input directly from repetition, despite having only rough knowledge of the actual system. Specifically, we propose the use of functional iterative learning control (fILC) [24], an ILC variant that is applicable to non-square systems, as opposed to standard ILC. The planning and control strategy is tested in simulation and validated on hardware using the parallel elastic quadruped Delft E-Go [7]. We show that our approach results in energy-efficient tracking of successive strides and stable forward locomotion. To constrain the complexity of the problem, this work is limited to the sagittal plane and only considers pronking gaits.

II. List of acronyms

ASR	articulated soft robot
CoM	center of mass
CoT	cost of transport
DoF	degree of freedom
EoM	equations of motion
fILC	functional iterative learning control
ILC	iterative learning control
MAE	mean absolute error
NLP	nonlinear program
PEA	parallel elastic actuator
SRBD	single rigid-body dynamics
SLIP	spring-loaded inverted pendulum
TD	touchdown
TO	take-off
TSLIP	trunk spring-loaded inverted pendulum

III. Controller overview

An overview of the proposed planning and control framework is presented in Figure 1. The framework consists of two main components; an offline trajectory planner and a learning controller. The trajectory planner generates an energy-efficient reference motion based on a 2D sagittal SRBD model. The learning controller learns the required control input to control a 3D full-order dynamics model, and overcomes the model discrepancy to track the reference produced by the planner.

IV. Model

A. Full-order model

The full-order dynamics of the E-go quadruped including spring forces can be described using a floating-base description [7],

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})+\mathbf{G}(\boldsymbol{q})+\mathbf{S}^{\mathsf{T}}\mathbf{K}(\boldsymbol{q}_{\mathrm{j}}-\boldsymbol{q}_{\mathrm{j}_0}) = \mathbf{S}^{\mathsf{T}}\boldsymbol{\tau}+\mathbf{J}_{\mathrm{c}}^{\mathsf{T}}\mathbf{F}_{\mathrm{c}} \quad (1)$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ includes the Coriolis and centrifugal terms, $\mathbf{G}$ includes the gravitational terms, $\mathbf{S}$ is the selection matrix of actuated joints, $\mathbf{K}$ is the spring stiffness matrix, $\boldsymbol{q}_{\mathrm{j}_0}$ is the vector of spring rest positions, $\boldsymbol{\tau}$ is the generalized torques vector, $\mathbf{J}_{\mathrm{c}}$ is the contact Jacobian and $\mathbf{F}_c$ are the contact forces. The generalized coordinates $\boldsymbol{q}$ of the E-Go quadruped consist of the actuated joint coordinates $\boldsymbol{q}_{\mathrm{j}} \in \mathbb{R}^{12}$ and unactuated base coordinates $\boldsymbol{q}_{\mathrm{b}} \in \mathbb{R}^{6}$, where the latter can only be controlled through the contact forces $\mathbf{F}_{\mathrm{c}}$, such that $\boldsymbol{q} = \begin{bmatrix}\boldsymbol{q}_{\mathrm{b}} & \boldsymbol{q}_{\mathrm{j}}\end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{18}$. Although the full-order floating-base model gives a very accurate description of the real system, planning trajectories can be very complex and computationally intensive due to the combination of a high number of degrees of freedom (DoFs) and nonlinear hybrid dynamics.

B. Reduced-order model

The number of DoFs can be reduced to make the problem of generating energy-efficient gaits more tractable, while still capturing the governing dynamics using the theory of templates and anchors [25]. A widely used template model for legged locomotion is the spring-loaded inverted pendulum (SLIP) model, which is known to very accurately describe the center of mass (CoM) motion of humans and animals [25]. Specifically, we select a variant of the standard SLIP model, the trunk spring-loaded inverted pendulum (TSLIP) model, resulting in additional pitch dynamics and an extra rotational DoF around the trunk CoM. The governing forces of the TSLIP model are the ground reaction force and the torque at the trunk. The TSLIP template behavior is then embedded into a more realistic anchor model whose morphology is closer to that of the actual system, with the addition of legs, actuators, and elastic joints, as depicted in Figure 2. The governing forces of the anchor model are still that of the template, except the forces are now generated in joint space. Hence, the governing behavior of the template and the anchor is equal.

The resulting model used in the trajectory optimization problem is a hybrid reduced-order SRBD model, considering only the 2D dynamics in the sagittal plane. The mass is

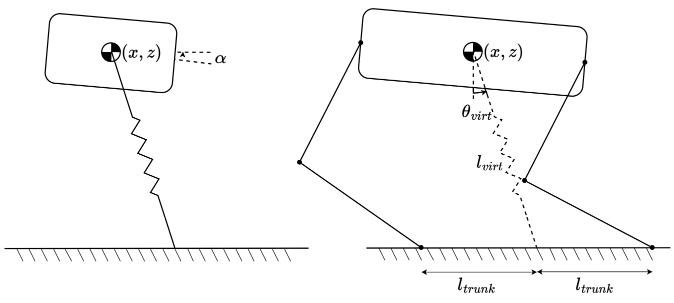


Fig. 2: Visual representation of the TSLIP template and SRBD anchor models. At touchdown, the hind and front feet positions of the anchor model are parameterized through the virtual TSLIP model. The virtual foot position is found using trigonometry and displaced in positive and negative x-direction with a distance equal to l_{trunk}, resulting in the hind and front feet positions.

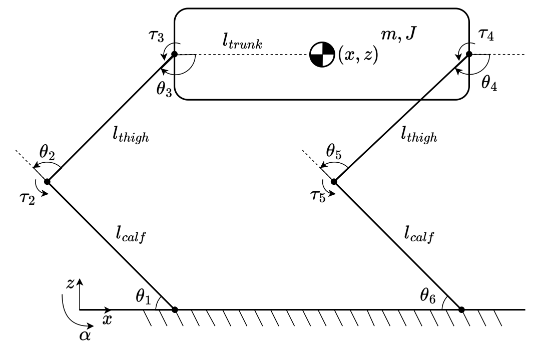


Fig. 3: Visual representation of the sagittal SRBD quadruped model used in the trajectory optimization, with massless legs and PEAs at the thigh and calf joints. As visualized here, the springs are currently in their rest positions.

concentrated in the trunk, while the legs are considered to be massless with parallel elastic actuators (PEAs) in the thigh and calf joints. The configuration of the robot can be represented by the Special Euclidean Group $SE(2)$, parameterized by the generalized coordinates $\boldsymbol{q} = \begin{bmatrix} x & z & \alpha \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$, where $\begin{bmatrix} x & z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$ is the position of the CoM of the trunk and α is its pitch angle. The system inputs are the motor torques $\boldsymbol{\tau} \in \mathbb{R}^4$ at the thigh and calf joints in joint coordinate space. A visual representation of this model is given in Figure 3.

1) Flight dynamics: In flight, the system follows a ballistic trajectory which can be modeled as

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{G}(\boldsymbol{q}) = \mathbf{0}, \tag{2}$$

where $\mathbf{M}$ is the mass matrix, and $\mathbf{G}$ includes the gravitational terms. Due to lack of contact between the feet and the ground, the system cannot be controlled while in this phase.

2) Stance dynamics: In stance, the generalized coordinates $\boldsymbol{q}$ are mapped to the joint coordinates $\boldsymbol{\theta} \in \mathbb{R}^6$ through the mapping $\boldsymbol{h} : \boldsymbol{q} \mapsto \boldsymbol{\theta}$, where $\boldsymbol{h}$ is obtained using inverse kinematics. In joint space, the contribution of the spring forces is computed. The equations of motion (EoM) are obtained using Lagrangian mechanics, resulting in the stance dynamics

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{G}(\boldsymbol{q}) + \mathbf{J_h}^\mathsf{T}(\boldsymbol{q})\mathbf{K}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_0\right) = \boldsymbol{\mathcal{F}}, \tag{3}$$

where $\mathbf{K}$ is the spring stiffness matrix, $\boldsymbol{\theta}_0$ is the spring rest positions, $\mathbf{J_h} = \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}\boldsymbol{q}}$ is the Jacobian that maps from joint space to generalized coordinates, and $\boldsymbol{\mathcal{F}}$ is the spatial wrench as a function of the motor torques $\boldsymbol{\tau}$. During the stance phase, the ground friction is assumed to be infinite, such that no slip occurs. The derivations of the mapping $\boldsymbol{h}$ and EoM are explained in Appendix A. Solving Equation 2 and Equation 3 for the accelerations $\ddot{\boldsymbol{q}}$ leads to

$$\begin{aligned} \ddot{\boldsymbol{q}} &= -\mathbf{M}^{-1}\mathbf{G} && \text{(flight)} \\ \ddot{\boldsymbol{q}} &= \mathbf{M}^{-1}\left(\boldsymbol{\mathcal{F}} - \mathbf{G} - \mathbf{J_h}^\mathsf{T}\mathbf{K}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_0\right)\right) && \text{(stance)}. \end{aligned} \tag{4}$$

The EoMs can then be rewritten to first-order ordinary differential equations, resulting in the general form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^6 \tag{5}$$

with the system state $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} & \dot{\boldsymbol{q}} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^6$ and control input $\boldsymbol{u} = \begin{bmatrix} \tau_2 & \tau_3 & \tau_4 & \tau_5 \end{bmatrix}^\mathsf{T} \in \mathbb{R}^4$.

C. Phase switching

In flight, the legs of the SRBD model do not follow a trajectory and are assumed to be able to move to any feasible position instantly, since the legs are massless. Consequently, this requires the determination of the feet positions at touchdown (TD). As both feet have to touch and leave the ground simultaneously, both feet positions can be parameterized at TD through the virtual TSLIP model, with the virtual touchdown leg angle θ_{virt} and leg length l_{virt} as its model parameters, as visualized in Figure 2. The virtual foot position $\boldsymbol{p}_{\text{virt}}$ in Cartesian coordinates is then a function of l_{virt} and θ_{virt}, given by $\boldsymbol{p}_{\text{virt}} = \begin{bmatrix} x + l_{\text{virt}}\sin(\theta_{\text{virt}}) & 0 \end{bmatrix}^\mathsf{T}$. The hind and front foot positions are then found by displacing $\boldsymbol{p}_{\text{virt}}$ in positive and negative x-direction with a distance equal to half the trunk length l_{trunk}. This gives two extra parameters to be selected, l_{virt} and θ_{virt}. The virtual leg length can be determined using the heuristic rule $l_{\text{virt}} = \sqrt{l_{\text{calf}}^2 + l_{\text{thigh}}^2}$, such that the calf springs are in their rest position at TD. The virtual leg angle is used as an input parameter to the optimization, and is proportional to the resulting forward velocity, as more forward velocity is required with increasing touchdown leg angle for dynamic stability.

V. Trajectory optimization

The optimization problem is formulated to find an energy-efficient periodic gait cycle, also known as a stride, by minimizing the cost of transport (CoT). A stride consists of three dynamic phases; an initial flight phase, a stance phase, and a final flight phase. The phases are separated by two events; a TD event, and a take-off (TO) event. We use a predefined contact sequence, where the two legs of the SRBD model are in flight simultaneously and in stance simultaneously, resulting in a pronking gait from the perspective of the sagittal plane. This reduces the complexity of the optimization problem, while maintaining the possibility to mirror the gait in the sagittal plane to produce trot or pace gaits for four legs.

The optimization problem is transcribed using multiple shooting, which divides the interval over which a solution is sought in $N-1$ intervals and N grid points. An initial value problem is solved on each of the $N-1$ intervals, and additional constraints are imposed such that the solution of adjacent intervals match at the grid points, forming a solution for the whole interval. The TD and TO events are handled by predefining the grid points n_{TD} and n_{TO} on which the respective events take place, where $0 < n_{\text{TD}} < n_{\text{TO}} < N$. The time step h is an optimization variable, such that the TD, TO, and final apex timings are optimization results and do not have to be specified a priori.

A. Objective function

The objective of the optimization problem is to find an energy-efficient periodic gait, by minimizing the CoT. The CoT is a commonly used metric to quantify the energy efficiency of different gaits. As it is a dimensionless quantity, it can be used to compare any form of locomotion of land, water, and airborne animals and vehicles. The CoT is defined as the energy input required to move a system of weight mg over a distance d. Both terms are expressed in mechanical work, where the energy input to the system is defined as the absolute mechanical work delivered by the actuators, to account for both positive and negative mechanical work [26]. It can be formulated as

$$\text{CoT} = \frac{E}{mgd} = \frac{\sum_{n=0}^{N-1} \sum_{i=2}^{5} \left| \tau_i \dot{\theta}_i \right|}{mgx_N}, \tag{6}$$

where x_N is the longitudinal position on the final node N.

B. Constraints

1) Initial condition and periodicity constraints: The stride starts at an apex, i.e. at zero vertical velocity, and at zero initial horizontal position

$$\begin{bmatrix} x_0 \\ \dot{z}_0 \end{bmatrix} = \mathbf{0}. \tag{7}$$

The remaining states of $\boldsymbol{x}_0$ are optimization variables and are found by the optimization program. The periodicity of the stride is enforced by constraining the final state $\boldsymbol{x}_N$ to be equal to the initial state $\boldsymbol{x}_0$ for all states in $\boldsymbol{x}$, except for the longitudinal position x, formulated as

$$\boldsymbol{x}_0 \backslash \{x_0\} = \boldsymbol{x}_N \backslash \{x_N\}. \tag{8}$$

2) Dynamic constraints: The system dynamics of the SRBD model is integrated at each grid point using fourth-order Runge-Kutta with the dynamics formulation from Equation 4 and Equation 5. A constraint is set at each grid point, to impose the continuity between the adjacent $N-1$ intervals. It is formulated as

$$\begin{aligned} \boldsymbol{x}_{n+1} &= \boldsymbol{F}(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n)) \\ &= \boldsymbol{x}_n + \frac{h}{6}(\boldsymbol{k_1} + 2\boldsymbol{k_2} + 2\boldsymbol{k_3} + \boldsymbol{k_4}) \\ \text{with } \boldsymbol{k}_1 &= \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(\boldsymbol{x}_n + \frac{h}{2}\boldsymbol{k_1}, \boldsymbol{u}_n) \\ \boldsymbol{k}_3 &= \boldsymbol{f}(\boldsymbol{x}_n + \frac{h}{2}\boldsymbol{k_2}, \boldsymbol{u}_n) \\ \boldsymbol{k}_4 &= \boldsymbol{f}(\boldsymbol{x}_n + h\boldsymbol{k_3}, \boldsymbol{u}_n). \end{aligned} \tag{9}$$

The transitions between the dynamic phases occur on the prespecified nodes n_{TD} and n_{TO}, such that the system is in flight before n_{TD} and after n_{TO}, and in stance between n_{TD} and n_{TO}, according to

$$\boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n) = \begin{cases} \boldsymbol{f}_{\text{flight}}(\boldsymbol{x}_n, \boldsymbol{u}_n) & \forall n \in [0, n_{\text{TD}}) \cup [n_{\text{TO}}, N] \\ \boldsymbol{f}_{\text{stance}}(\boldsymbol{x}_n, \boldsymbol{u}_n) & \forall n \in [n_{\text{TD}}, n_{\text{TO}}) \end{cases}. \tag{10}$$

3) Switch conditions: Switching between the dynamic phases requires extra constraints at the switching nodes to guarantee dynamic feasibility. As the front and hind feet are parameterized through the virtual TSLIP model, the system is said to touch or leave the ground whenever the virtual foot enters or leaves ground contact. A TD event can then be formulated as the moment when the system state is in the touchdown manifold, given by

$$\mathcal{X}_{\text{TD}} = \{\boldsymbol{x} \mid z - l_{\text{virt}} \cos(\theta_{\text{virt}}) = 0,\ \dot{z} < 0\}. \tag{11}$$

Similarly, a TO event occurs when the system state is in the take-off manifold, given by

$$\mathcal{X}_{\text{TO}} = \{\boldsymbol{x} \mid \sqrt{x^2 + z^2} - l_{\text{virt}} = 0,\ \dot{z} > 0\}. \tag{12}$$

The system is then constrained to be in the TD and TO manifold at n_{TD} and n_{TO} respectively, such that $\mathbf{x}_{n_{\text{TD}}} \in \mathcal{X}_{\text{TD}}$ and $\mathbf{x}_{n_{\text{TO}}} \in \mathcal{X}_{\text{TO}}$.

C. Problem formulation

The trajectory optimization problem is formulated as a nonlinear program (NLP). Using the aforementioned objective function and constraints, the full optimization problem is formulated as

$$
\begin{aligned}
\min_{\substack{\boldsymbol{x}_0,\dots,\boldsymbol{x}_N \\ \boldsymbol{u}_0,\dots,\boldsymbol{u}_{N-1} \\ h}} \quad & \frac{\sum_{n=0}^{N-1}\sum_{i=2}^{5}\left|\tau_i\dot{\theta}_i\right|}{mgx_N} \\
\text{s.t.} \quad & \begin{bmatrix} x_0 & \dot{z}_0 \end{bmatrix}^T = \mathbf{0} \\
& \boldsymbol{x}_0 \backslash \{x_0\} = \boldsymbol{x}_N \backslash \{x_N\} \\
& \boldsymbol{x}_{n+1} = \boldsymbol{F}(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n)) \; \forall n \in [0, N-1] \\
\boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n) = & \begin{cases} \boldsymbol{f}_{\text{flight}}(\boldsymbol{x}_n, \boldsymbol{u}_n), & \forall n \in [0, n_{\text{TD}}) \cup [n_{\text{TO}}, N] \\ \boldsymbol{f}_{\text{stance}}(\boldsymbol{x}_n, \boldsymbol{u}_n), & \forall n \in [n_{\text{TD}}, n_{\text{TO}}) \end{cases} \\
& \boldsymbol{x}_{n_{\text{TD}}} \in \mathcal{X}_{\text{TD}} \\
& \boldsymbol{x}_{n_{\text{TO}}} \in \mathcal{X}_{\text{TO}} \\
& \boldsymbol{x}_{\min} \leq \boldsymbol{x}_{\text{n}} \leq \boldsymbol{x}_{\max} \\
& \boldsymbol{u}_{\min} \leq \boldsymbol{u}_{\text{n}} \leq \boldsymbol{u}_{\max} \\
& h_{\min} \leq h \leq h_{\max}
\end{aligned}
\tag{13}
$$

VI. Trajectory tracking

To track the reference produced by the planner, the reduced-order model behavior has to be mapped to the full-order system. To achieve this, we propose the use of fILC to overcome the model mismatch and iteratively learn the feedforward control signal required to track the reference.

A. Functional Iterative Learning Control

1) Background: Originally, fILC was developed for linear systems [24]. Hence, first consider the linear continuous system

$$
\begin{aligned}
\dot{\boldsymbol{x}}_j(t) &= \mathbf{A}\boldsymbol{x}_j(t) + \mathbf{B}\boldsymbol{u}_j(t) \\
\boldsymbol{y}_j(t) &= \mathbf{C}\boldsymbol{x}_j(t),
\end{aligned}
\tag{14}
$$

with iteration index j, $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{B} \in \mathbb{R}^{n\times l}, \mathbf{C} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x}_j \in \mathbb{R}^n$, $\boldsymbol{u}_j \in \mathbb{R}^l$, $\boldsymbol{y}_j \in \mathbb{R}^m$, and $l < m$, meaning that the system is underactuated. There are two main differences that differentiate fILC from standard ILC:

1. fILC does not track the reference $\bar{\boldsymbol{y}}$ completely, but rather tracks it at certain time instances of interest, given by $\{T^1, \dots, T^o\}$, where o is the number of time instances and the superscript denotes the index. The desired output of the system at the time instances is denoted as $\{\bar{\boldsymbol{y}}^1 \dots \bar{\boldsymbol{y}}^o\}$, given by the planner. The goal of the controller then is to iteratively learn a control input $\boldsymbol{u}_j(t)$, such that

$$
\lim_{j\to\infty} \boldsymbol{y}_j(T^k) = \bar{\boldsymbol{y}}^k, \quad \forall k \in 1, \dots, o. \tag{15}
$$

2. The control input $\boldsymbol{u}_j(t)$ is not learned directly, but it is learned in a functional subspace of continuous basis functions $\boldsymbol{\pi}$. Sampling the control input from a large enough subspace makes fILC applicable to highly non-square systems. The continuous basis functions provide an infinite amount of degrees of freedom, making the problem square [24]. The control input is therefore parameterized as a linear combination of functions, given by

$$
\begin{aligned}
\boldsymbol{u}_j(t) &= \boldsymbol{\pi}(t)\boldsymbol{\alpha}_j, \\
\boldsymbol{\pi} &= \begin{bmatrix} \boldsymbol{\pi}^1 \dots \boldsymbol{\pi}^o \end{bmatrix} \in \mathbb{R}^{l\times mo},
\end{aligned}
\tag{16}
$$

where $\boldsymbol{\pi}$ is a matrix of basis functions, with $\boldsymbol{\pi}^i(t) \in \mathbb{R}^{l\times m}$ and weight vector $\boldsymbol{\alpha}_j \in \mathbb{R}^{mo}$. The closed form solution of Equation 14 is

$$
\boldsymbol{y}_j(t) = \mathbf{C}e^{\mathbf{A}t}\boldsymbol{x}(0) + \mathbf{C}\int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\boldsymbol{u}_j(\tau)\mathrm{d}\tau. \tag{17}
$$

Substituting Equation 16 and sampled at the i-th time instance, results in

$$
\boldsymbol{y}_j\left(T^i\right) = \mathbf{C}e^{\mathbf{A}T^i}\boldsymbol{x}(0) + \left(\int_0^{T^i} \mathbf{C}e^{\mathbf{A}\left(T^i-\tau\right)}\mathbf{B}\boldsymbol{\pi}(\tau)\mathrm{d}\tau\right)\boldsymbol{\alpha}_j. \tag{18}
$$

Subsequently, this can be written in super-vector notation for all o time instances, according to

$$
\mathbf{Y}_j = \boldsymbol{d}_j + \mathbf{H}\boldsymbol{\alpha}_j, \tag{19}
$$

where $\boldsymbol{d}_j \in \mathbb{R}^{mo}$ is the free response and $\mathbf{H} \in \mathbb{R}^{mo\times mo}$ is the forced response to the basis functions $\boldsymbol{\pi}$. fILC now has a form equivalent to that of standard discrete ILC [27]. Therefore, by learning the control input in functional space, the learning task is reduced to learning a discrete set of weights $\boldsymbol{\alpha}_j$, while the input $\boldsymbol{u}_j$ remains continuous. This also eliminates the need to discretize the system and control input, as is typically a requirement for standard ILC due to its discrete nature. The weights $\boldsymbol{\alpha}_j$ are learned iteratively through proportional error feedback, given a typical ILC learning rule

$$
\begin{aligned}
\boldsymbol{\alpha}_{j+1} &= \boldsymbol{\alpha}_j + \mathbf{L}\mathbf{E}_j \\
&= \boldsymbol{\alpha}_j + \mathbf{L}\left(\bar{\mathbf{Y}} - \mathbf{Y}_j\right) \\
&= \boldsymbol{\alpha}_j + \mathbf{L}\begin{bmatrix} \bar{\boldsymbol{y}}^1 - \boldsymbol{y}_j\left(T^1\right) \\ \vdots \\ \bar{\boldsymbol{y}}^o - \boldsymbol{y}_j\left(T^o\right) \end{bmatrix},
\end{aligned}
\tag{20}
$$

where $\mathbf{L} \in \mathbb{R}^{mo\times mo}$ is the learning gain, $\mathbf{E}_j$ is the iteration error, $\bar{\mathbf{Y}}$ is the reference, and $\mathbf{Y}_j$ is the iteration output, with the latter three in super-vector notation. The fILC system is asymptotically stable, with $\boldsymbol{x}_j(0) = \boldsymbol{x}_0 \; \forall \; j$, if and only if

$$
\rho(\mathbf{I} - \mathbf{L}\mathbf{H}) < 1, \tag{21}
$$

where ρ is the spectral radius.

2) Nonlinear systems: Now consider the full-order continuous nonlinear quadruped system, with unknown exact dynamics of the form

$$
\begin{aligned}
\dot{\boldsymbol{x}}_j(t) &= \boldsymbol{f}(t, \boldsymbol{x}_j(t), \boldsymbol{u}_j(t)) \\
\boldsymbol{y}_j(t) &= \boldsymbol{g}(t, \boldsymbol{x}_j(t), \boldsymbol{u}_j(t)),
\end{aligned}
\tag{22}
$$

with iteration index j, $\boldsymbol{x}_j \in \mathbb{R}^n$, $\boldsymbol{u}_j \in \mathbb{R}^l$, $\boldsymbol{y}_j \in \mathbb{R}^m$, and $l < m$. The main difference between linear and nonlinear fILC is that in the nonlinear case, $\mathbf{H}$ cannot be obtained as in Equation 18. As $\mathbf{H}$ is the input-output map of the system as a response to the basis functions $\boldsymbol{\pi}$, sampled at the time instances $\{T^1, \dots, T^o\}$, it can be formulated for nonlinear systems as

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{g}(T^1, \boldsymbol{x}, \boldsymbol{\pi}^1) & \boldsymbol{g}(T^1, \boldsymbol{x}, \boldsymbol{\pi}^2) & \cdots & \boldsymbol{g}(T^1, \boldsymbol{x}, \boldsymbol{\pi}^{mo}) \\ \boldsymbol{g}(T^2, \boldsymbol{x}, \boldsymbol{\pi}^1) & \boldsymbol{g}(T^2, \boldsymbol{x}, \boldsymbol{\pi}^2) & \cdots & \boldsymbol{g}(T^2, \boldsymbol{x}, \boldsymbol{\pi}^{mo}) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{g}(T^o, \boldsymbol{x}, \boldsymbol{\pi}^1) & \boldsymbol{g}(T^o, \boldsymbol{x}, \boldsymbol{\pi}^2) & \cdots & \boldsymbol{g}(T^o, \boldsymbol{x}, \boldsymbol{\pi}^{mo}) \end{bmatrix} \in \mathbb{R}^{mo \times mo}. \tag{23}$$

The matrix $\mathbf{H}$ can be obtained from experiments, by exciting the system from an equilibrium and recording the responses, omitting the need to derive an explicit model. A block diagram of the controller is depicted in Figure 1.

B. Controller design

The learning gain $\mathbf{L}$ and basis functions $\boldsymbol{\pi}$ provide two design choices for the controller.

1) Learning rule: We use a linear quadratic learning rule to compute the optimal learning gain $\mathbf{L}$, that minimizes the following cost function

$$\mathbf{J}(\boldsymbol{\alpha}_j) = \|\mathbf{E}_j\|^2_{\mathbf{Q}_{\text{LQ}}} + \|\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_{j-1}\|^2_{\mathbf{S}_{\text{LQ}}}, \tag{24}$$

such that

$$\mathbf{L} = \left(\mathbf{H}^{\mathsf{T}}\mathbf{Q}_{\text{LQ}}\mathbf{H} + \mathbf{S}_{\text{LQ}}\right)^{-1}\mathbf{H}^{\mathsf{T}}\mathbf{Q}_{\text{LQ}}, \tag{25}$$

where $\mathbf{Q}_{\text{LQ}}$ and $\mathbf{S}_{\text{LQ}}$ are diagonal gain matrices [27]. $\mathbf{Q}_{\text{LQ}}$ and $\mathbf{S}_{\text{LQ}}$ can then be tuned to penalize the error and the change in control input respectively.

2) Basis functions: As the control input is parameterized by a linear combination of functions, the selection of an appropriate family of basis functions provides an important design parameter to design the behavior of the controller. Although any choice of π suffices such that $\mathbf{H}$ is full rank [24], the properties of the functions should be taken into account, as they will be propagated into the control input. For example, sinusoidal basis functions will result in an oscillatory control input.

We select a set of mo Gaussians as our basis functions, with the mean μ and variance σ^2 per Gaussian to select. As the robot can only be controlled during the stance phase, the means are distributed evenly between the TD and TO timings, t_{TD} and t_{TO} respectively, and given equal variance. Outside this interval they are set to zero, formulated as

$$\pi^i(t) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-\mu^i}{\sigma}\right)^2} & \text{if } t_{\text{TD}} \leq t \leq t_{\text{TO}} \\ 0 & \text{otherwise} \end{cases}. \tag{26}$$

The configuration of the basis functions π^i in the basis function matrix $\boldsymbol{\pi}$ should also be considered to prevent coupling of the control inputs, as each column of $\boldsymbol{\pi}$ is multiplied with a single scalar weight α^i. To prevent this, each column of $\boldsymbol{\pi}$ should only contain one basis function. Furthermore, each row of $\boldsymbol{\pi}$ should preferably have an approximate equal amount of basis functions, such that the control authority is distributed equally over the number of available control inputs. We use the following configuration for $\boldsymbol{\pi}$,

$$\boldsymbol{\pi}(t) = \begin{bmatrix} \boldsymbol{\pi}^1 & \ldots & \boldsymbol{\pi}^o \end{bmatrix} = \begin{bmatrix} \pi^1(t) & 0 & 0 & 0 & \pi^5(t) & 0 & \cdots & 0 \\ 0 & \pi^2(t) & 0 & 0 & 0 & \pi^6(t) & \cdots & 0 \\ 0 & 0 & \pi^3(t) & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \pi^4(t) & 0 & 0 & \cdots & \pi^{mo}(t) \end{bmatrix}. \tag{27}$$

C. Continuous learning

In addition to the discontinuous learning process inherent to ILC, where the system is reset at the end of each iteration, we have investigated using fILC in a continuous approach without resetting the system after each iteration. In fact, this approach is more akin to repetitive control [28], where the final state of the current iteration is used as the initial state for the next iteration. We modify the learning rule from Equation 20 and add a diagonal scaling matrix $\mathbf{W}$ to increase the learning rate in some states, and to switch it off in others. Most importantly, learning is switched off in forward x-direction, to allow the robot to move forward, as the reference itself does not change per iteration. The new learning rule becomes

$$\boldsymbol{\alpha}_{j+1} = \boldsymbol{\alpha}_j + \mathbf{L}(\mathbf{W}\mathbf{E}_j), \tag{28}$$

where $\mathbf{W} \in \mathbb{R}^{mo \times mo}$. We then omit the system reset at the end of each iteration. Aside from the aforementioned changes, the controller design is left unchanged. It should be noted that we only provide experimental validation of this method, while this work lacks theoretical guarantees due to lack of time.

VII. Simulations and experiments

A. Software setup

The proposed planning and control framework is tested and validated in simulation. The trajectory planner and controller are both implemented in Python. The open-source nonlinear optimization framework CasADi [29] is used to formulate the trajectory optimization problem, which uses algorithmic differentiation to obtain the derivatives for gradient-based optimization and interfaces with IPOPT [30] to solve it. We use the rigid-body simulator PyBullet [31] to simulate the full-order system.

B. Implementation

The parameters of the reduced-order model used in the trajectory optimization are based on the E-Go quadruped [7]. The mass m and inertia J represent the total mass and total inertia of the real robot. The rest spring angles are set as visualized in Figure 3. The spring stiffnesses are selected such that the simulated robot in PyBullet is able to sustain its own weight without the use of its actuators while in standstill, which was found to be $50\,\mathrm{Nm/rad}$ for all calf and thigh joints. To correctly represent the real system, which has twice the amount of legs, the 2D model should produce twice the amount of torque per leg. Therefore, the spring stiffnesses of the 2D model are set at $100\,\mathrm{Nm/rad}$. The virtual leg length is set at $l_{\text{virt}} = \sqrt{l^2_{\text{calf}} + l^2_{\text{thigh}}} = 0.301\,\mathrm{m}$. The virtual leg touchdown

TABLE I: Optimization model parameters

Parameter	Symbol		Value
mass	m	(kg)	12.013
inertia	J	$(\mathrm{kg\,m^2})$	0.103
calf length	l_{calf}	(m)	0.213
thigh length	l_{thigh}	(m)	0.213
half trunk length	l_{trunk}	(m)	0.188
virtual leg length	l_{virt}	(m)	0.301
virtual leg angle	θ_{virt}	$(\frac{\pi}{180}$ rad)	13
hind calf spring rest angle	θ_{0_2}	$(\pi/180°\text{rad})$	45
hind thigh spring rest angle	θ_{0_3}	$(\pi/180°\text{rad})$	-135
front calf spring rest angle	θ_{0_4}	$(\pi/180°\text{rad})$	-135
front thigh spring rest angle	θ_{0_5}	$(\pi/180°\text{rad})$	45
hind calf spring stiffness	k_2	(Nm/rad)	100
hind thigh spring stiffness	k_3	(Nm/rad)	100
front calf spring stiffness	k_4	(Nm/rad)	100
front thigh spring stiffness	k_5	(Nm/rad)	100

angle θ_{virt} is set at $13\,°$. All used model parameters and their values are presented in Table I.

The number of grid points of the optimization problem is set at $N = 150$, yielding a good balance between solving time and solution accuracy. The three dynamic phases are spaced evenly over the grid and have an equal number of nodes, i.e. 50 nodes per phase, such that touchdown occurs at node $n_{\text{TD}} = 50$ and take-off at node $n_{\text{TD}} = 100$. The flight apexes occur at the initial node n_0 and final node N. Analogous to the spring stiffnesses, the torque limits of the actuators are twice that of the real system. The optimization parameters are summerized in Table II.

TABLE II: Optimization parameters

Parameter	Symbol		Value
number of grid points	N	(-)	150
touchdown node	n_{TD}	(-)	50
take-off node	n_{TO}	(-)	100
time step limits	h_{min}, h_{min}	(s)	0.001, 0.01
forward position limits	$x_{\text{min}}, x_{\text{max}}$	(m)	0, 2
vertical position limits	$z_{\text{min}}, z_{\text{max}}$	(m)	0, 2
angular position limits	$\alpha_{\text{min}}, \alpha_{\text{max}}$	(m)	0, 5
forward velocity limits	$\dot{x}_{\text{min}}, \dot{x}_{\text{max}}$	(m/s)	0, 5
vertical velocity limits	$\dot{z}_{\text{min}}, \dot{z}_{\text{max}}$	(m/s)	-2, 2
angular velocity limits	$\dot{\alpha}_{\text{min}}, \dot{\alpha}_{\text{max}}$	(m)	0, 2
hind calf actuator limits	$\tau_{2_{\text{min}}}, \tau_{2_{\text{max}}}$	(N m)	-71.1, 71.1
hind thigh actuator limits	$\tau_{3_{\text{min}}}, \tau_{3_{\text{max}}}$	(N m)	-47.4, 47.4
front thigh actuator limits	$\tau_{4_{\text{min}}}, \tau_{4_{\text{max}}}$	(N m)	-47.4, 47.4
front thigh actuator limits	$\tau_{5_{\text{min}}}, \tau_{5_{\text{max}}}$	(N m)	-71.1, 71.1

For the functional iterative learning controller, the selected time instances of interest are the lowest point of the trajectory, the take-off event and the final apex. As the time step h and, as a consequence, the total stride time are optimization results, the selected time instances are described as the node indices $\{75, 100, 150\}$. At the start of the learning process, the system is set to the initial state determined by the planner and the weights of the first iterations are set to zero, such that $\boldsymbol{\alpha}_0 = \mathbf{0}$. After an iteration has finished, the system is reset to the initial state and the next control signal is executed. The mean absolute error (MAE) is used to compare the tracking error of fILC, as it measures performance irrespective of the used number of time instances. It is defined as

$$\text{MAE} = \frac{\sum_{i=1}^{mo} \left| \bar{\mathbf{Y}}_i - \mathbf{Y}_{j_i} \right|}{mo}. \tag{29}$$

C. Analysis

To analyze the performance of the proposed planning and control approach, the results of the planner and controller will be compared under two scenarios, with and without springs. For the controller, the feedforward control signal produced by fILC will be compared directly with the control input signal found by the trajectory optimization. These experiments will show the benefit of elastic joints in terms of energy efficiency, and that the controller is required to overcome the model mismatch and to track the given reference.

For the continuous learning approach, the method is tested under two conditions. The controller will first learn the weights to successfully traverse flat terrain. Subsequently, the learned controller is transferred to an environment with randomly generated uneven terrain, to test the robustness of the controller.

D. Hardware setup

We validate the method of continuous fILC on hardware using the E-Go quadruped. This requires the input-output map from Equation 23 to be determined, for which the system response to $mo = 6 \cdot 3 = 18$ Gaussians has to be recorded. Obtaining the mapping on hardware is a tedious process, which has to be redone every time the parameters of the Gaussians are changed. Therefore, we omit this by recording the mapping in simulation and transferring the resulting $\mathbf{H}$ matrix to the hardware directly.

The springs of the E-Go quadruped are much softer than the ones used in the PyBullet simulation, namely $6\,\mathrm{N\,m}$ for the calf joints and $16\,\mathrm{N\,m}$ for the thigh joints, compared to $50\,\mathrm{N\,m}$ used for all joints in simulation. As a result, the physical springs are not stiff enough to sustain the weight of the E-Go quadruped without additional motor input. We compensate for the softer springs using a PD-impedance controller, acting as a virtual spring in parallel to the physical spring, according to

$$\boldsymbol{\tau}_{\text{PD}} = \mathbf{K}_{\text{P}} \left(\boldsymbol{q}_{\text{ref}} - \boldsymbol{q}_{\text{j}} \right) - \mathbf{K}_{\text{D}} \left(\dot{\boldsymbol{q}}_{\text{j}} \right), \tag{30}$$

where $\boldsymbol{q}_{\text{j}}$ are the joint positions, $\boldsymbol{q}_{\text{ref}}$ the joint reference positions, $\dot{\boldsymbol{q}}_{\text{j}}$ the joint velocities, $\mathbf{K}_{\text{P}}$ is the proportional gain and $\mathbf{K}_{\text{D}}$ is the derivative gain. The joint reference positions $\boldsymbol{q}_{\text{ref}}$ are equal to the spring rest positions $\boldsymbol{q}_{\text{j}_0}$, such that the combined torque of the springs and the PD-impedance controller is roughly similar to the spring torque from simulation. Given the full-order EoM from Equation 1, the total input torque $\boldsymbol{\tau}$ sent to the motors is equal to the sum of the fILC input $\boldsymbol{u}_{\text{j}}$ and the impedance controller compensation $\boldsymbol{\tau}_{\text{PD}}$. The used gains are $K_{\text{P}} = 60$ and $K_{\text{D}} = 1$ for all joints.

It should be noted that the state estimator of the robot is not reliable in our case, as only the actuator positions are used to determine the robot state. This might be sufficient for gaits when at least one leg is in contact with the ground, but this is not the case for our pronking gait where all four feet are off the ground simultaneously during the flight phase, resulting in inaccurate state estimates. Despite this, the largest inaccuracies are observed in x-direction, which is not used in the continuous learning approach.

VIII. Results

A. Trajectory optimization

Besides the model and optimization parameters from Table I and Table II, the trajectory optimization has the touchdown leg angle θ_{virt} as the only input parameter to shape the output of the optimization, which is expected to be proportional to the forward velocity. We can validate this by varying the virtual leg angle from 5° to 30°, with increments of 1°, and plotting it against the average forward velocity $\dot{x}_{\text{avg}}$, as shown in Figure 4. A linear relationship between the virtual leg angle and the average forward velocity is observed.

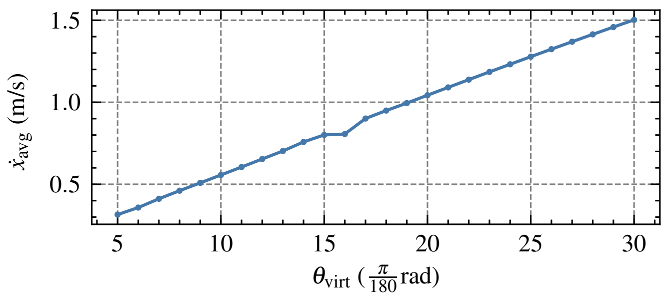


Fig. 4: Average forward velocity $\dot{x}_{\text{avg}}$ as a function of the virtual touchdown leg angle θ_{virt}.

For the stiff case, the optimization finds the optimal solution in approximately 9 s, resulting in a CoT of 0.596. With elastic joints, the optimization program solves in approximately 7 s, with a CoT of 0.563. The results of both optimizations are shown in Figure 5 and Figure 6 respectively. We observe very similar trajectories between the states of both cases, in terms of shape and magnitude. Only a larger decrease in forward velocity is observed during the stance phase for the stiff case. The most noticeable difference is the resulting control input. Without springs, the control inputs are strictly negative, whereas with springs, both thigh joints produce positive torque. Overall, the torque magnitudes are lower for the elastic case than for the stiff case.

B. Trajectory tracking

The optimal control inputs found for both the stiff and elastic case are fed into the full-order system directly without learning. For both cases, large errors are reported in all states, with an MAE of 0.456 for the stiff case and 0.350 for the elastic case. The resulting CoTs are 2.83 and 3.83 respectively. The full state evolutions for both cases are presented in Appendix B.

Now we let the controller find the control inputs to track the given reference. Figure 7 shows how the controller iteratively learns to track the reference points, and that the MAE decreases rapidly and converges after 150 iterations. The final MAE is 9.83×10^{-3}, with a measured CoT of 0.765. The whole learning process takes approximately 4 s. The used controller parameters are $\mathbf{Q}_{\text{LQ}} = \mathbf{S}_{\text{LQ}} = \mathbf{I}_{18}$ and Gaussian variance $\sigma^2 = 0.0004$. The tracking result in all system states, as well as the learned weights and resulting control input are presented in Figure 8. It shows that the controller is able to

TABLE III: Comparison of planned CoT and resulting CoT and MAE, for both the stiff and elastic case. Bold indicates lowest per category.

	Planner		**Controller**		
	stiff	elastic	stiff without learning	elastic without learning	elastic with fILC
CoT	0.596	**0.563**	2.83	3.83	**0.765**
MAE	-	-	0.456	0.350	**0.00983**

track most desired outputs with high precision, with small discrepancies observed in the angular position and forward velocity, and with motor torques that are within actuator limits. The CoT and MAE for all cases are summarized in Table III.

In a similar fashion, it is also possible to track multiple successive strides. The results of tracking a double stride are reported in Appendix B for the sake of space.

C. Continuous learning

We apply our continuous learning approach with the scaling factors $s = (0, 10, 1, 2, 2, 1)$, $\mathbf{W} = \text{diag}(s, s, s)$, gains $\mathbf{Q}_{\text{LQ}} = 0.05\mathbf{I}_{18}$ and $\mathbf{S}_{\text{LQ}} = \mathbf{I}_{18}$, and Gaussian variance $\sigma^2 = 0.001$. The time indices are left unchanged. A more "jumpy" reference is used to make the effect of the controller clearer. The results after 110 iterations are presented in Figure 9 and Figure 10. Figure 10 shows the evolution of the position over time. Starting from an equilibrium, we observe that the controller learns to track the reference trajectory resulting in a stable forward pronking gait. Figure 9 shows that the MAE decreases gradually as the number of iterations increases, resulting in a final CoM trajectory that looks very similar to the reference. It took the robot approximately 25 s to cover the distance in Figure 10.

The robustness of the learned controller is tested on randomly generated uneven terrain, by initializing the weights of a new learning cycle with the previously learned weights on flat terrain. The remaining controller parameters are left unchanged. The result is presented in Figure 10. We observe that the controller is able to traverse the uneven terrain despite the disturbance, with a CoM trajectory that still looks relatively similar to the reference. Initialized with the weights from the previous cycle, it took the robot approximately 5 s to cover the same distance.

D. Hardware experiments

1) Flat terrain: The continuous learning approach is validated on hardware. Initially, the controller is tuned without connecting the springs, as the springs have to be detached and reconnected every time the robot restarts, such as when it falls over. Connecting the springs can be a time-consuming and tedious process. Therefore, the controller parameters are first tuned without springs, resulting in the scaling factors $s = (0, 10, 1, 2, 2, 1)$, $\mathbf{W} = \text{diag}(s, s, s)$, gains $\mathbf{Q}_{\text{LQ}} = 0.1\mathbf{I}_{18}$ and $\mathbf{S}_{\text{LQ}} = \mathbf{I}_{18}$, and Gaussian variance $\sigma^2 = 0.1$. These controller parameters are used to learn the weights on flat terrain without the springs connected.

Thereafter, these learned weights are transferred and used to initialize the weights for a new learning cycle with the springs

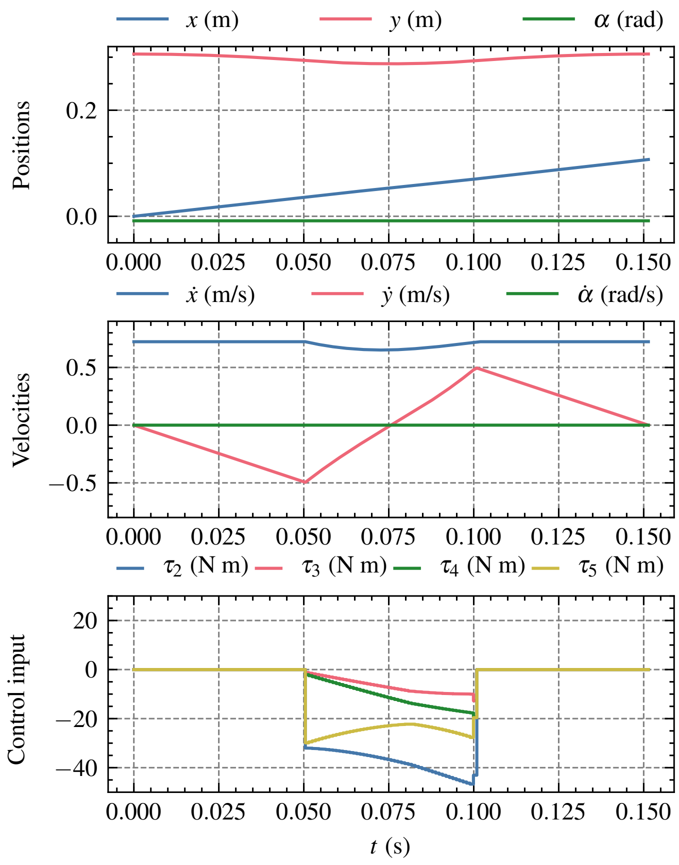


Fig. 5: Trajectory optimization result without springs.

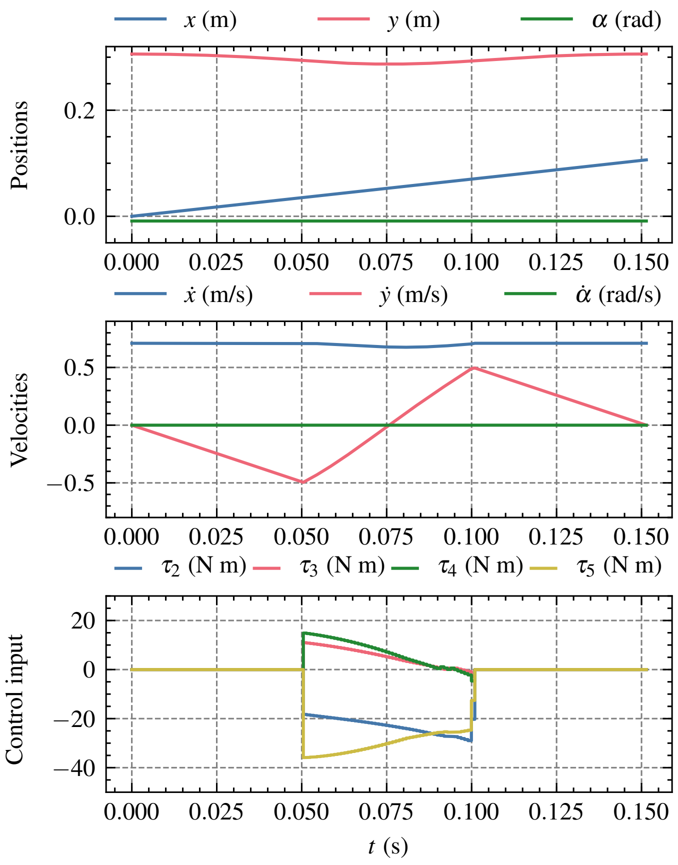


Fig. 6: Trajectory optimization result with springs.

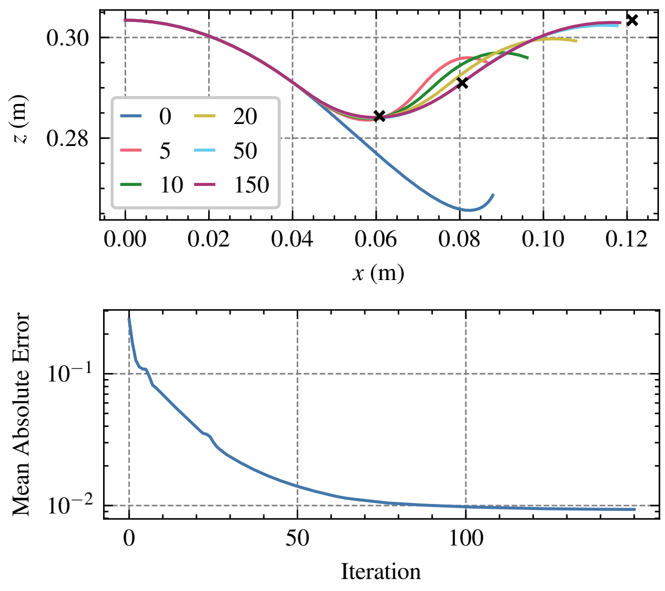


Fig. 7: Visualization of the fILC learning process. The top panel shows the CoM trajectory tracking result in x and z-coordinates at specific iteration numbers, indicated by the legend. The black crosses indicate the desired state output at a time instance. The bottom panel shows the evolution of the tracking performance as a function of the number of iterations. The final MAE after 150 iterations is 9.83×10^{-3}.

connected. The learning gain $\mathbf{Q}_{\text{LQ}}$ is decreased to $0.01\mathbf{I}_{18}$. We start from standstill and let the robot learn until it runs out of the $3\,\text{m}$ of physical space to move, which occurs after $52\,\text{s}$. At the end of this learning cycle, we again transfer the learned weights to a new learning cycle using the same controller parameters. This time, it takes the robot $20\,\text{s}$ to reach the end of the testing space. Figure 11 shows the result of both subsequent learning cycles and the learned control input of the final iteration. Figure 12 presents several snapshots of the learning process.

2) Uneven terrain: We use the same methodology of transferring weights, and validate the robustness of the learned controller on uneven terrain outdoors, using the learned weights from the previous indoor learning cycle and the same controller parameters. The robot is placed on concrete bricks while facing grassy ground. We then start the new learning cycle. We observe that the controller is able to successfully transition between different types of terrain, transitioning from rigid concrete to uneven grassy ground. Figure 12 shows several snapshots of the transitions between terrain.

3) Incline: Finally, the controller is tested on an incline of $5°$. Again, the cycle is initialized with the weights learned indoors, using the same controller parameters. We let the robot face the incline, starting from level ground. The controller successfully transitions from the level ground to the incline. Several snapshots of the transition are presented in Figure 12.

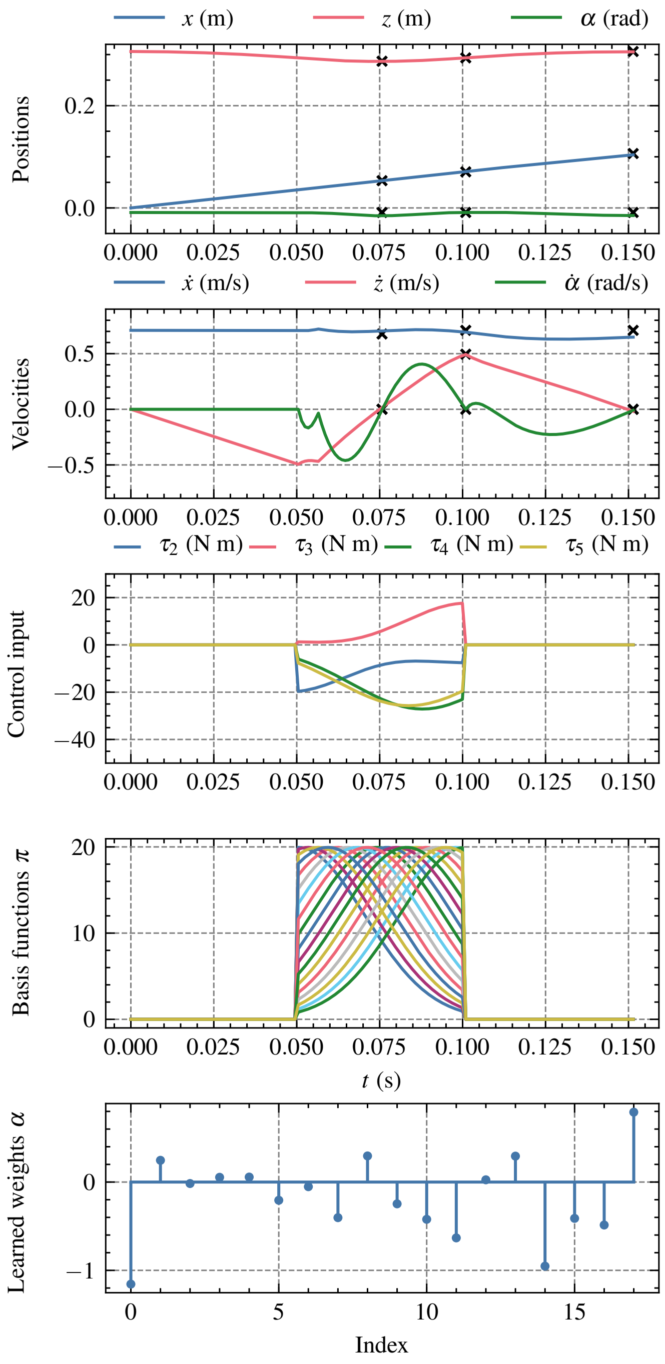


Fig. 8: Evolution of the system state, control input and basis functions after 150 iterations, CoT=0.765. The bottom panel shows the values of the learned weights. The black crosses indicate the desired state output at a time instance.

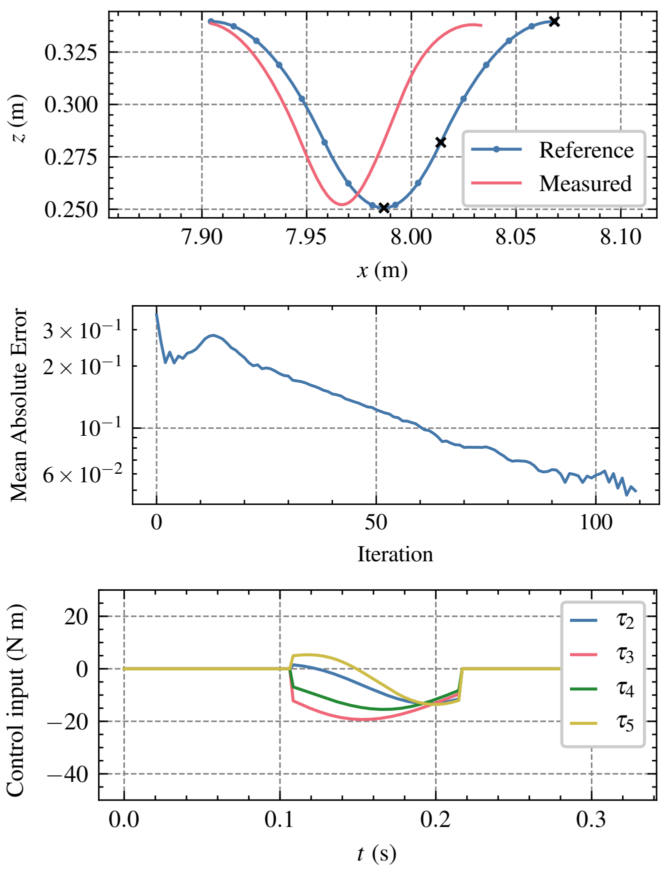


Fig. 9: Continuous learning with fILC. The top panel shows the resulting CoM trajectory after 110 iterations and 8 m traveled in forward direction. The center panel shows the evolution of the MAE as a function of the number of iterations. The bottom panel shows the learned control input at the final iteration.

IX. Discussion

A. Trajectory optimization

Figure 4 shows the positive linear relationship between the touchdown leg angle θ_{virt} and the average forward velocity $\dot{x}_{\text{avg}}$, confirming that the virtual touchdown angle is proportional to the forward velocity. Hence, it can be concluded that the touchdown leg angle is an appropriate input parameter to the trajectory optimization. The optimization results, as summarized in Table III, confirm that compared to stiff locomotion, elastic elements improve the walking efficiency, as suggested by [32].

B. Trajectory tracking

From the results presented in Table III, it is evident that the optimal control signals do not result in the desired reference output when fed forward into the full-order system without learning. Therefore, it can be concluded that the model discrepancy between the used reduced-order and the full-order model is too large to use the optimization output directly, and that a controller is required to overcome the model mismatch. From Figure 8, we find that fILC is very effective at accomplishing this task, resulting in seemingly perfect tracking in most states. The large change in angular velocity observed during the stance phase is the result of only tracking certain time instances instead of the whole reference. Whereas the time instances are successfully tracked, the controller has no knowledge of what happens at other moments in time. This, in combination with the forward momentum, results in the observed change in pitch angle. Regardless, the high tracking accuracy shows that the controller is able to overcome

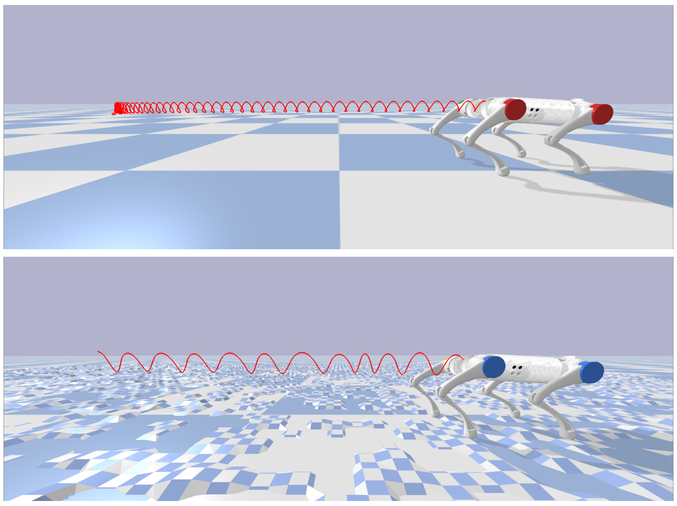


Fig. 10: Continuous learning with fILC in PyBullet simulation. The top figure shows the controller is able to learn to track the reference from standstill, without resetting the system at the end of each iteration. The bottom figure shows the result of transferring the weights learned on flat terrain to rough terrain. The red trajectory indicates the evolution of the CoM over time.

and learn the unmodeled dynamics, including the effect of ground contact collision, ground friction, joint friction, spring damping, and leg mass and inertia.

In terms of energy efficiency, Table III shows that the resulting CoT is an order of magnitude higher without the use of fILC, whereas the magnitudes of the control inputs are of the same order. A possible explanation could be that the observed vertical and angular velocities of the CoM during the stance phase without fILC are much higher, leading to higher joint velocities and as a result, a higher CoT. Compared to the planned CoT, the resulting CoT is in the same order of magnitude, albeit slightly higher. This is to be expected, as the planned CoT does not account for leg mass and inertia, ground contact collision and friction forces. Similarly, the higher angular velocity also contributes to a higher CoT, compared to the absence of rotational velocity observed in the planned trajectory.

C. Continuous learning

Despite yielding accurate tracking of single strides, repeating the learned control input in a feedforward manner does not result in stable locomotion, diverging after several successive strides. Therefore, feedforward control by itself is not sufficient for stable locomotion and additional feedback is required to stabilize the gait. It should be noted that this divergence is a property of feedforward control in general, not of the proposed method. The continuous learning approach provides a method of achieving stable locomotion, without altering the stiffness behavior through feedback, resulting in a forward pronking motion that is very similar to the reference. Furthermore, when transferring over the learned weights from flat terrain, the controller is able to confidently traverse the rough ground. Despite violating the identical initialization condition of ILC, which states that each iteration should start at the same state, the controller seems robust to the large initialization errors. A thorough explanation of this cannot be given and would require more analysis, although it has been shown before that ILC can be robust to initialization errors [33].

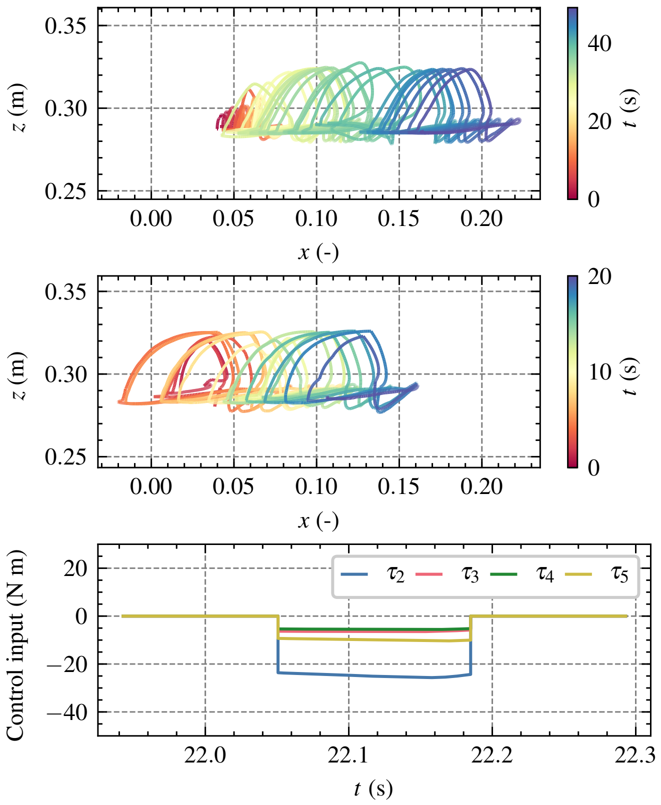


Fig. 11: Two subsequent learning cycles on hardware with springs connected, where the second cycle is initialized with the weights learned in the first cycle. The top panel shows the evolution in position, initialized with the weights learned on hardware without springs. The center panel shows the evolution of the subsequent learning cycle, initialized with the weights of the previous cycle. The color gradient indicates the evolution of time. Note that the x-axis has no unit due to the state estimation inaccuracy. The bottom panel shows the control input learned in the final iteration of the last cycle.

D. Hardware experiments

The validation of the continuous learning approach on hardware proves that this method is not only limited to simulation. Using the input-output map from simulation, the proposed method is able to learn a forward pronking gait directly on hardware in only several minutes. Furthermore, the resulting controller is robust to different types of terrain and inclines, and transitions between them effortlessly. Again, it should be noted that the state estimates of the robot are inaccurate, most noticeably in x-position and x-velocity, reporting a distance traveled of $0.2\,\mathrm{m}$, whereas the robot covered approximately $3\,\mathrm{m}$ in reality during the indoor experiments. As a result, the controller perceives an error that is higher than the true error, which could lead to instability as the number of iterations increases. As demonstrated, this can be prevented by initially

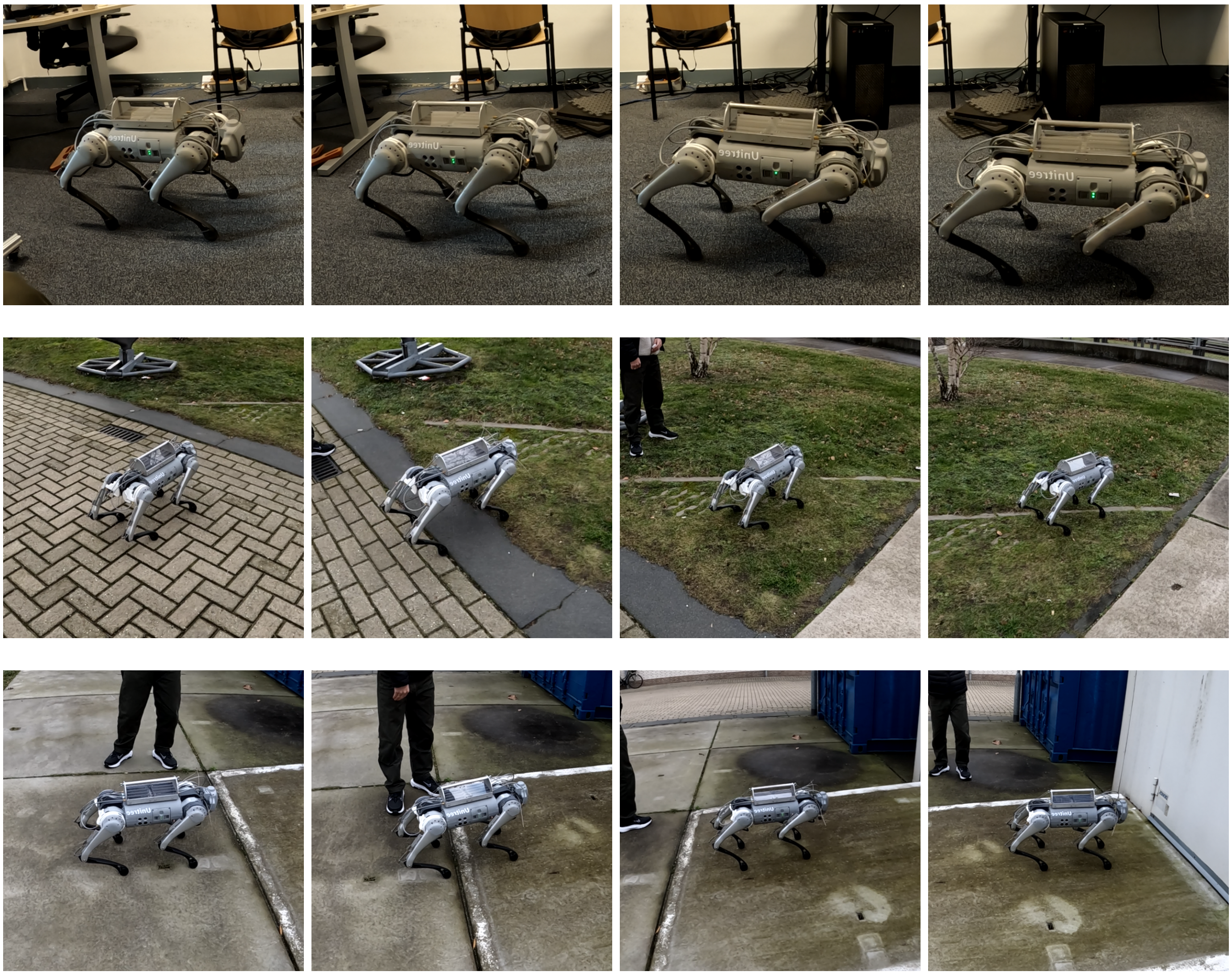


Fig. 12: Hardware experiments of continuous fILC on, from top to bottom, flat terrain, rough grassy terrain, and on an incline. The robot is able to successfully transition between different types of terrain.

learning a set of weights with a higher gain until the motion is acceptable, decreasing the gain, and transferring the weights to a new learning cycle.

E. Future work

Several recommendations can be made to improve the performance of the proposed methods. Regarding fILC, different learning rules could be investigated that, for example, incorporate more model knowledge or put more emphasis on minimization of control input, potentially increasing the convergence rate, tracking accuracy, or energy efficiency. An example of this could be gradient-based learning rules [34]. Different gaits such as the pace and the trot gait could be incorporated for a wider variety of motions, although this will be significantly more difficult due to the inherent instability of these dynamic gaits. It is likely that lateral control would be necessary to successfully execute these gaits, which as a result would extend the work beyond the sagittal plane. Currently, the controller has no knowledge of the legs of the quadruped. Therefore, learning to track joint states in addition to the base state could improve the tracking performance for successive strides. In turn, this would require the generation of joint trajectories, for example using full-body optimization, which could be an improvement of the trajectory planner. Another approach for improved execution of successive strides could be the use of feedback control in parallel with fILC, called current-iteration ILC [35]. Using small feedback gains, the system could apply small corrections to stabilize the motion during the stride and limit the initialization error, while changing the stiffness behavior only marginally.

Regarding the continuous learning approach, more research is required to analyze the mechanisms behind this method and to formulate theoretical guarantees. For the performed experiments, an obvious point of improvement is the use of more accurate state estimates, either through the incorporation of additional sensor data from an inertial measurement unit or vision sensors, or via an external positioning systems.

X. Conclusion

This work proposes a motion planning and control strategy for efficient and compliant single stride locomotion using trajectory optimization and fILC. We have demonstrated that our approach is able to learn the feedforward control signal required to make an articulated soft quadruped track a reference with high accuracy and efficiency, in only a matter of seconds. It achieves this while preserving the physical compliance of the system and without reliance on an accurate mathematical description of the full-order system. As a consequence, this work validates that fILC is applicable to hybrid nonlinear underactuated systems. The proposed strategy is not necessarily restricted to locomotion, but is applicable to ASRs in general. This work also demonstrates that fILC can be used in a continuous approach, resulting in a pronking gait that is learned on hardware in minutes. To the best of the author's knowledge, this work is the first to use ILC to control a quadruped without the use of additional feedback controllers, and the first to apply ILC to compliant legged systems in general. As only experimental validation of the continuous approach is given, future work will include the formulation of theoretical guarantees.

References

[1] C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world applications for legged robots," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1311–1326, 12 2018.

[2] Boston Dynamics, "Spot Specifications," 2022. [Online]. Available: https://support.bostondynamics.com/s/article/Robot-specifications

[3] ANYbotics, "ANYmal - Autonomous Legged Robot," 2022. [Online]. Available: https://www.anybotics.com/anymal-autonomous-legged-robot/

[4] T. J. Roberts and E. Azizi, "Flexible mechanisms: the diverse roles of biological springs in vertebrate movement," *Journal of Experimental Biology*, vol. 214, no. 3, pp. 353–361, 2 2011. [Online]. Available: https://journals.biologists.com/jeb/article/214/3/353/33503/Flexible-mechanisms-the-diverse-roles-of

[5] G. A. Cavagna, "Force platforms as ergometers," *https://doi.org/10.1152/jappl.1975.39.1.174*, vol. 39, no. 1, pp. 174–179, 1975. [Online]. Available: https://journals.physiology.org/doi/abs/10.1152/jappl.1975.39.1.174

[6] C. Della Santina, M. G. Catalano, and A. Bicchi, "Soft Robots," *Encyclopedia of Robotics*, pp. 1–15, 2021.

[7] J. Ding, P. Posthoorn, V. Atanassov, J. Kober, and C. D. Santina, "Delft E-Go Compliant Quadrupedal Robot: Parallel Compliance Design, Locomotion Control, and Hardware Experiments," 2023.

[8] C. David, C. David Remy, K. Buffinton, and R. Siegwart, "Energetics of passivity based running with high-compliance series elastic actuation ETH Library Energetics of passivity based running with high-compliance series elastic actuation," 2010. [Online]. Available: https://doi.org/10.3929/ethz-a-010027887

[9] C. D. Remy, "Optimal exploitation of natural dynamics in legged locomotion," 2011. [Online]. Available: https://doi.org/10.3929/ethz-a-6665065

[10] Z. Gan and C. D. Remy, "A passive dynamic quadruped that moves in a large variety of gaits," in *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., 10 2014, pp. 4876–4881.

[11] W. Xi and C. D. Remy, "Optimal gaits and motions for legged robots," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3259–3265, 10 2014.

[12] W. Xi, Y. Yesilevskiy, and C. David Remy, "Selecting gaits for economical locomotion of legged robots," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1140–1154, 2016.

[13] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 5, pp. 783–792, 10 2010.

[14] A. Werner, R. Lampariello, and C. Ott, "Trajectory optimization for walking robots with series elastic actuators," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2015-February, no. February, pp. 2964–2970, 2014.

[15] C. Della Santina, M. Bianchi, G. Grioli, F. Angelini, M. Catalano, M. Garabini, and A. Bicchi, "Controlling Soft Robots: Balancing Feedback and Feedforward Elements," *IEEE Robotics and Automation Magazine*, vol. 24, no. 3, pp. 75–83, 9 2017.

[16] M. Keppler, D. Lakatos, C. Ott, and A. Albu-Schaffer, "Elastic Structure Preserving (ESP) Control for Compliantly Actuated Robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 317–335, 4 2018.

[17] A. De Luca and F. Flacco, "Dynamic gravity cancellation in robots with flexible transmissions," *Proceedings of the IEEE Conference on Decision and Control*, pp. 288–295, 2010.

[18] D. Lakatos, C. Rode, A. Seyfarth, and A. Albu-Schäffer, "Design and control of compliantly actuated bipedal running robots: Concepts to exploit natural system dynamics," *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-February, pp. 930–937, 2 2015.

[19] M. Hutter, C. D. Remy, M. A. Hoepflinger, and R. Siegwart, "Efficient and versatile locomotion with highly compliant legs," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 2, pp. 449–458, 2013.

[20] M. Hutter, C. Gehring, M. Bloesch, M. Hoepflinger, P. Fankhauser, and R. Siegwart, "Excitation and stabilization of passive dynamics in locomotion using hierarchical operational space control," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., 9 2014, pp. 2977–2982.

[21] G. M. Gasparri, S. Manara, D. Caporale, G. Averta, M. Bonilla, H. Marino, M. Catalano, G. Grioli, M. Bianchi, A. Bicchi, and M. Garabini, "Efficient Walking Gait Generation via Principal Component Representation of Optimal Trajectories: Application to a Planar Biped Robot with Elastic Joints," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2299–2306, 7 2018.

[22] M. J. Pollayil, C. D. Santina, G. Mesesan, J. Englsberger, D. Seidel, M. Garabini, C. Ott, A. Bicchi, and A. Albu-Schaffer, "Planning Natural Locomotion for Articulated Soft Quadrupeds," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 6593–6599, 2022.

[23] T. Apgar, P. Clary, K. Green, A. Fern, and J. Hurst, "Fast Online Trajectory Optimization for the Bipedal Robot Cassie," *Robotics: Science and Systems*, 2018.

[24] C. Della Santina and F. Angelini, "Iterative Learning in Functional Space for Non-Square Linear Systems," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2021-December, pp. 5858–5863, 2021.

[25] R. J. Full and D. E. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, 12 1999. [Online]. Available: https://journals.biologists.com/jeb/article/202/23/3325/8334/Templates-and-anchors-neuromechanical-hypotheses

[26] H. C. Doets, D. Vergouw, H. E. Veeger, and H. Houdijk, "Metabolic cost and mechanical work for the step-to-step transition in walking after successful total ankle arthroplasty," *Human Movement Science*, vol. 28, no. 6, pp. 786–797, 12 2009.

[27] R. J. Li and Z. Z. Han, "Survey of iterative learning control," *Kongzhi yu Juece/Control and Decision*, vol. 20, no. 9, pp. 961–966, 9 2005.

[28] G. Hillerström and K. Walgama, "Repetitive Control Theory and Applications - A Survey," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 1446–1451, 6 1996.

[29] 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, no. 1, pp. 1–36, 3 2019. [Online]. Available: https://link.springer.com/article/10.1007/s12532-018-0139-4

[30] 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, no. 1, pp. 25–57, 5 2006.

[31] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2021. [Online]. Available: https://pybullet.org/

[32] S. Manara, G. M. Gasparri, M. Garabini, D. Caporale, M. Gabiccini, and A. Bicchi, "Analysis of series elasticity in locomotion of a planar bipedal robot," *International Journal of Mechanics and Control*, vol. 20, no. 01, 2019.

[33] K. Hu, C. Ott, and D. Lee, "Online iterative learning control of zero-moment point for biped walking stabilization," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June. Institute of Electrical and Electronics Engineers Inc., 6 2015, pp. 5127–5133.

[34] T. Sogo and N. Adachi, “Iterative learning control based on the gradient method for linear discrete-time systems,” *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 4729–4734, 7 1999.

[35] T. Y. Doh, J. H. Moon, K. B. Jin, and M. J. Chung, “Robust iterative learning control with current feedback for uncertain linear systems,” *http://dx.doi.org.tudelft.idm.oclc.org/10.1080/002077299292650*, vol. 30, no. 1, pp. 39–47, 2010. [Online]. Available: https://www-tandfonline-com.tudelft.idm.oclc.org/doi/abs/10.1080/002077299292650

[36] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. T. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Å. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, “SymPy: Symbolic computing in python,” *PeerJ Computer Science*, vol. 2017, no. 1, p. e103, 1 2017. [Online]. Available: https://peerj.com/articles/cs-103

APPENDIX A
EQUATIONS OF MOTION

A. Inverse kinematics

At touchdown, the position of the TSLIP virtual foot $\boldsymbol{p}_{\text{virt}}$ is given by

$$\boldsymbol{p}_{\text{virt}} = \begin{bmatrix} x_{\text{virt}} \\ z_{\text{virt}} \end{bmatrix} = \begin{bmatrix} x + l_{\text{virt}} \sin(\theta_{\text{virt}}) \\ 0 \end{bmatrix} \tag{31}$$

The virtual foot position $\boldsymbol{p}_{\text{virt}}$ is then displaced in positive and negative x direction by a distance equal to the half trunk length l_{trunk}, resulting in the hind and front foot positions, as visualized in Figure 2

$$\boldsymbol{p}_{\text{foot}_\text{H}} = \boldsymbol{p}_{\text{virt}} - \begin{bmatrix} l_{\text{trunk}} \\ 0 \end{bmatrix}, \; \boldsymbol{p}_{\text{foot}_\text{F}} = \boldsymbol{p}_{\text{virt}} + \begin{bmatrix} l_{\text{trunk}} \\ 0 \end{bmatrix}. \tag{32}$$

The positions of the hind and front hip are described as

$$\boldsymbol{p}_{\text{hip}_\text{H}} = \begin{bmatrix} x \\ z \end{bmatrix} - \begin{bmatrix} l_{\text{trunk}} \cos(\alpha) \\ l_{\text{trunk}} \sin(\alpha) \end{bmatrix}, \; \boldsymbol{p}_{\text{hip}_\text{F}} = \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} l_{\text{trunk}} \cos(\alpha) \\ l_{\text{trunk}} \sin(\alpha) \end{bmatrix}. \tag{33}$$

Finally, the hip positions are translated to the origin, treating $\boldsymbol{p}_{\text{hip}}$ as the base and $\boldsymbol{p}_{\text{foot}}$ as the end effector, according to

$$\boldsymbol{p}_e = \boldsymbol{p}_{\text{foot}} - \boldsymbol{p}_{\text{hip}} = \begin{bmatrix} x_e \\ z_e \end{bmatrix}. \tag{34}$$

Geometric inverse kinematics is then used to calculate the joint positions, resulting in the mapping $\boldsymbol{h} : \boldsymbol{q} \mapsto \boldsymbol{\theta}$ that maps the generalized coordinates to joint coordinates. The definition of the joint angles is visualized in Figure 3 and given by

$$\begin{aligned}
\theta_2 &= \cos^{-1}\left(\frac{x_{e_H}^2 + z_{e_H}^2 - l_{\text{calf}}^2 - l_{\text{thigh}}^2}{2 l_{\text{calf}} l_{\text{thigh}}}\right) \\
\theta_3 &= \tan^{-1}\left(\frac{z_{e_H}}{x_{e_H}}\right) - \tan^{-1}\left(\frac{l_{\text{thigh}} \sin(\theta_2)}{l_{\text{calf}} + l_{\text{thigh}} \cos(\theta_2)}\right) \\
\theta_1 &= -\theta_2 - \theta_3 \\
\theta_5 &= \cos^{-1}\left(\frac{x_{e_F}^2 + z_{e_F}^2 - l_{\text{calf}}^2 - l_{\text{thigh}}^2}{2 l_{\text{calf}} l_{\text{thigh}}}\right) \\
\theta_4 &= \tan^{-1}\left(\frac{z_{e_F}}{x_{e_F}}\right) - \tan^{-1}\left(\frac{l_{\text{thigh}} \sin(\theta_5)}{l_{\text{calf}} + l_{\text{thigh}} \cos(\theta_5)}\right) \\
\theta_6 &= -\theta_5 - \theta_4
\end{aligned} \tag{35}$$

B. Stance dynamics

Given the mapping $\mathbf{h}$, the EoM of the quadruped in stance can be computed using Lagrangian mechanics:

$$\begin{aligned}
T &= \frac{1}{2} m \left(\dot{x}^2 + \dot{z}^2\right) + \frac{1}{2} J \dot{\alpha}^2 \\
V &= mgz + \frac{1}{2} k_2 \left(\theta_{0_2} - \theta_2\right)^2 + \frac{1}{2} k_3 \left(\theta_{0_3} - \theta_3\right)^2 + \frac{1}{2} k_4 \left(\theta_{0_4} - \theta_4\right)^2 \\
&\quad + \frac{1}{2} k_5 \left(\theta_{0_5} - \theta_5\right)^2 \\
\mathcal{L} &= T - V \\
&= \frac{1}{2} m \left(\dot{x}^2 + \dot{z}^2\right) + \frac{1}{2} J \dot{\alpha}^2 - \frac{1}{2} k_2 \left(\theta_{0_2} - \theta_2\right)^2 - \frac{1}{2} k_3 \left(\theta_{0_3} - \theta_3\right)^2 \\
&\quad - \frac{1}{2} k_4 \left(\theta_{0_4} - \theta_4\right)^2 - \frac{1}{2} k_5 \left(\theta_{0_5} - \theta_5\right)^2
\end{aligned} \tag{36}$$

Solving the Euler-Lagrange equation $\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\delta \mathcal{L}}{\delta \dot{\mathbf{q}}}\right) - \frac{\delta \mathcal{L}}{\delta \mathbf{q}}$ with $q = \begin{bmatrix} x & z & \alpha \end{bmatrix}^\mathsf{T}$ yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\delta \mathcal{L}}{\delta \dot{\boldsymbol{q}}}\right) = \begin{bmatrix} m\ddot{x} \\ m\ddot{z} \\ J\ddot{\alpha} \end{bmatrix}, \quad \frac{\delta \mathcal{L}}{\delta \boldsymbol{q}} = \begin{bmatrix} \sum_{i=2}^{5} k_i(\theta_{0_i} - \theta_i) \frac{\mathrm{d}}{\mathrm{d}x} \theta_i \\ \sum_{i=2}^{5} k_i(\theta_{0_i} - \theta_i) \frac{\mathrm{d}}{\mathrm{d}z} \theta_i - mg \\ \sum_{i=2}^{5} k_i(\theta_{0_i} - \theta_i) \frac{\mathrm{d}}{\mathrm{d}\alpha} \theta_i \end{bmatrix}, \tag{37}$$

resulting in the EoM

$$\begin{gathered}
\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \ddot{\alpha} \end{bmatrix} + \begin{bmatrix} \sum_{i=2}^{5} k_i(\theta_i - \theta_{0_i}) \frac{\mathrm{d}}{\mathrm{d}x} \theta_i \\ \sum_{i=2}^{5} k_i(\theta_i - \theta_{0_i}) \frac{\mathrm{d}}{\mathrm{d}z} \theta_i \\ \sum_{i=2}^{5} k_i(\theta_i - \theta_{0_i}) \frac{\mathrm{d}}{\mathrm{d}\alpha} \theta_i \end{bmatrix} + \begin{bmatrix} 0 \\ mg \\ 0 \end{bmatrix} = \begin{bmatrix} F_x \\ F_z \\ \tau_\alpha \end{bmatrix} \\
\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \left(\frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}\boldsymbol{q}}\right)^\mathsf{T} \mathbf{K}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_0\right) + \mathbf{G}(\boldsymbol{q}) = \boldsymbol{\mathcal{F}} \\
\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{J_h}^\mathsf{T}(\boldsymbol{q}) \mathbf{K}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_0\right) + \mathbf{G}(\boldsymbol{q}) = \boldsymbol{\mathcal{F}},
\end{gathered} \tag{38}$$

where $\mathbf{K} = \operatorname{diag}(0, k_1, k_2, k_3, k_4, k_5, 0)$ and $\boldsymbol{\mathcal{F}}$ is the spatial wrench. The partial derivatives of the joint angles with respect to $\mathbf{q}$ are obtained symbolically using SymPy [36]. The spatial wrench is given by

$$\boldsymbol{\mathcal{F}} = \begin{bmatrix} F_x \\ F_z \\ \tau_\alpha \end{bmatrix} = \begin{bmatrix} F_{\text{H}_x} + F_{\text{F}_x} \\ F_{\text{H}_z} + F_{\text{F}_z} \\ \boldsymbol{F}_\text{H} \times \boldsymbol{r}_\text{H} + \boldsymbol{F}_\text{F} \times \boldsymbol{r}_\text{F}, \end{bmatrix} \tag{39}$$

where $\boldsymbol{F}_\text{H}$ and $\boldsymbol{F}_\text{F}$ are the ground reaction forces of each leg as a result of the motor torque $\boldsymbol{\tau}$, and $\boldsymbol{r}_\text{H}$ and $\boldsymbol{r}_\text{F}$ are the vectors from the CoM to the foot contact points. The motor torques $\boldsymbol{\tau}$ are converted from joint coordinate space to generalized forces using the contact Jacobians of the hind and front leg, $\mathbf{J}_\text{H}$ and $\mathbf{J}_\text{F}$ respectively. As the legs exert a force on the ground, the ground reaction force that is exerted on the robot is equal and opposite, given by

$$\begin{gathered}
\boldsymbol{F} = -\left(\mathbf{J}^{-\mathsf{T}} \boldsymbol{\tau}\right) \\
\begin{bmatrix} \boldsymbol{F}_\text{H} \\ \boldsymbol{F}_\text{F} \end{bmatrix} = -\begin{bmatrix} \mathbf{J}_\text{H}^{-\mathsf{T}} & 0 \\ 0 & \mathbf{J}_\text{F}^{-\mathsf{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}_\text{H} \\ \boldsymbol{\tau}_\text{F} \end{bmatrix} \\
\begin{bmatrix} F_{\text{H}_x} \\ F_{\text{H}_z} \\ F_{\text{H}_\alpha} \\ F_{\text{F}_x} \\ F_{\text{F}_z} \\ F_{\text{F}_\alpha} \end{bmatrix} = -\begin{bmatrix} & & & 0 & 0 & 0 \\ & \mathbf{J}_\text{H}^{-\mathsf{T}} & & 0 & 0 & 0 \\ & & & 0 & 0 & 0 \\ 0 & 0 & 0 & & & \\ 0 & 0 & 0 & & \mathbf{J}_\text{F}^{-\mathsf{T}} & \\ 0 & 0 & 0 & & & \end{bmatrix} \begin{bmatrix} 0 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ 0 \end{bmatrix},
\end{gathered} \tag{40}$$

where the subscript denotes the ground reaction force component in the direction of one of the generalized coordinates of $\boldsymbol{q}$. As the joint torques are converted to equivalent forces at the feet, F_{H_α} and F_{F_α} are both equal to zero.

C. Contact Jacobians

The contact Jacobians of the front and hind leg are the following. The Jacobian of the front leg is different to the hind leg to accommodate for the specific order of torques in the torque vector $\boldsymbol{\tau} = \begin{bmatrix} 0 & \tau_2 & \tau_3 & \tau_4 & \tau_5 & 0 \end{bmatrix}^{\mathsf{T}}$.

1) Hind leg:

$$\mathbf{J}_{\mathrm{H}} = \begin{bmatrix} 0 & -l_{\text{calf}}\sin(\theta_3+\theta_2) & -l_{\text{thigh}}\sin(\theta_3) - l_{\text{calf}}\sin(\theta_3+\theta_2) \\ 0 & l_{\text{calf}}\cos(\theta_3+\theta_2) & l_{\text{thigh}}\cos(\theta_3) + l_{\text{calf}}\cos(\theta_3+\theta_2) \\ 1 & 1 & 1 \end{bmatrix} \tag{41}$$

2) Front leg:

$$\mathbf{J}_{\mathrm{F}} = \begin{bmatrix} -l_{\text{thigh}}\sin(\theta_4) - l_{\text{calf}}\sin(\theta_4+\theta_5) & -l_{\text{calf}}\sin(\theta_4+\theta_5) & 0 \\ l_{\text{thigh}}\cos(\theta_4) + l_{\text{calf}}\cos(\theta_4+\theta_5) & l_{\text{calf}}\cos(\theta_4+\theta_5) & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{42}$$

Appendix B
Trajectory tracking

A. *Without learning*

Figure 13 and Figure 14 depict the full state evolutions, as a response to the optimal controls inputs from optimization without learning, as mentioned in subsection VIII-B and subsection IX-B. Large errors are observed in all states without learning.

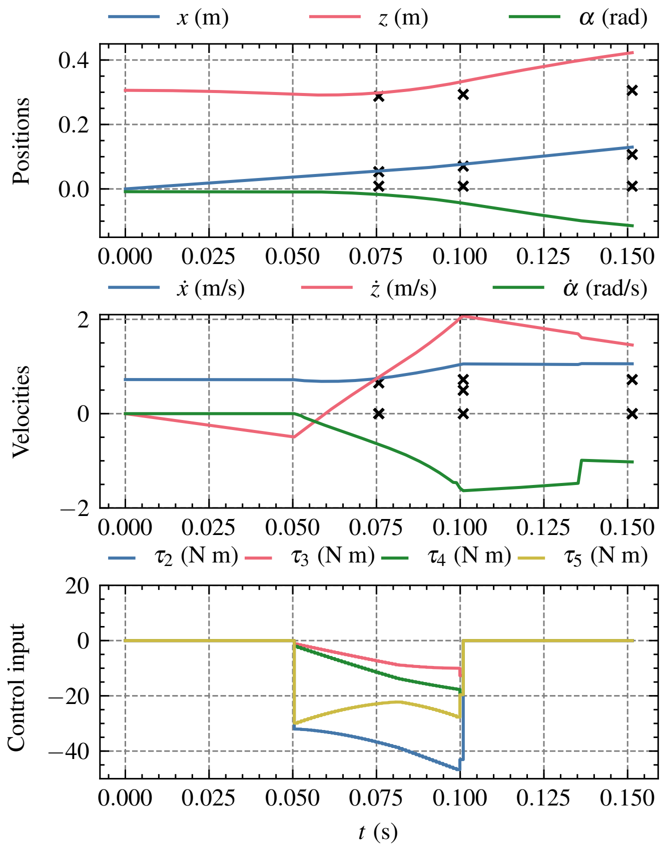


Fig. 13: Result of feedforward control signal from optimization without springs. The black crosses indicate the desired state output at a time instance.

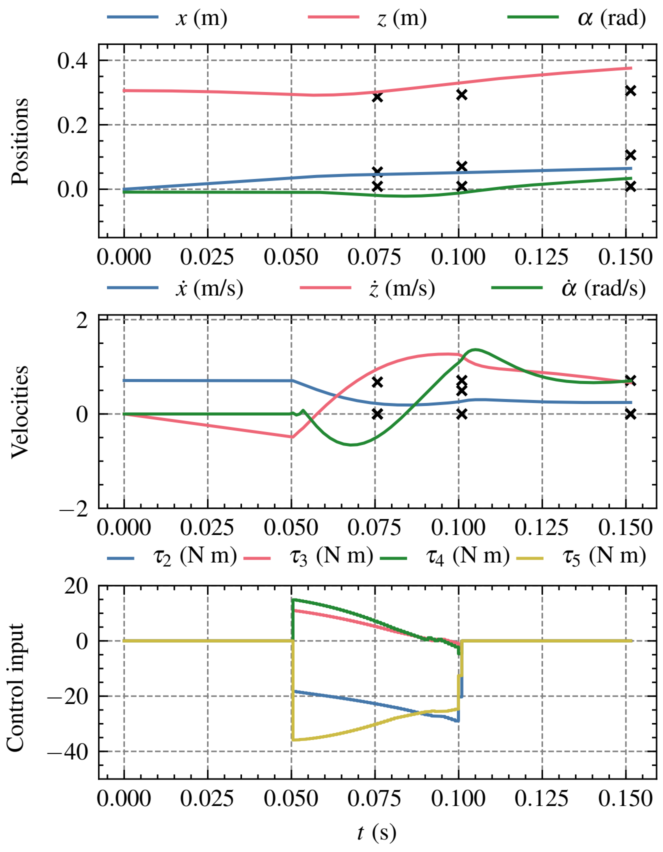


Fig. 14: Result of feedforward control signal from optimization with springs. The black crosses indicate the desired state output at a time instance.

B. Double stride

The following figures present the result of tracking a double stride. The single stride reference, including the time instances, is repeated in x-direction once, resulting in the time instances $\{50, 100, 150, 225, 250, 300\}$. Figure 15 shows the learning curve for tracking a double stride, reaching a MAE of 1.7×10^{-2} after 300 iterations. Figure 16 shows the resulting CoM trajectory. Again, we observe a deviation in tracking the lowest points of the trajectory at T^1 and T^4, while the apexes at T^3 and T^6 are tracked with high accuracy. Figure 17 shows the full state evolution. Similarly, the controller is able to track the desired state outputs with high precision, with the pitch angle α being the only exception, with an error of ± 0.1 rad.

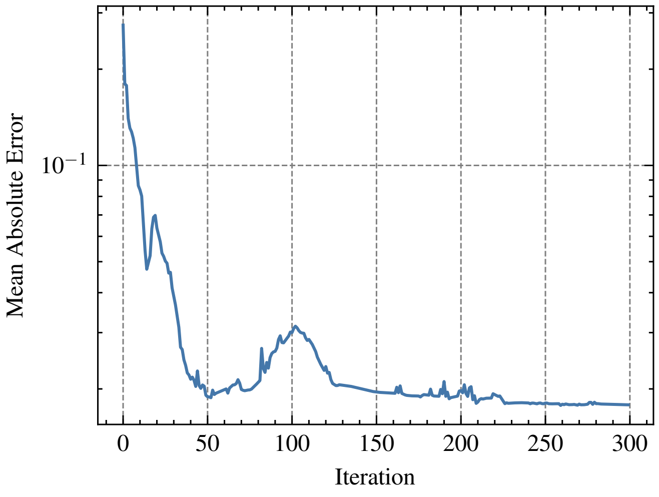


Fig. 15: Evolution of the MAE as a function of the number of iterations for tracking a double stride. The final MAE $= 1.7 \times 10^{-2}$.

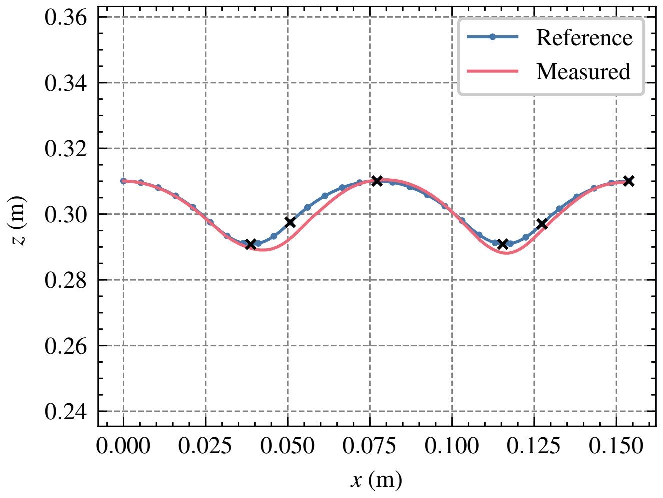


Fig. 16: Double stride CoM trajectory tracking result after 300 iterations. The black crosses indicate the desired state output at a time instance.

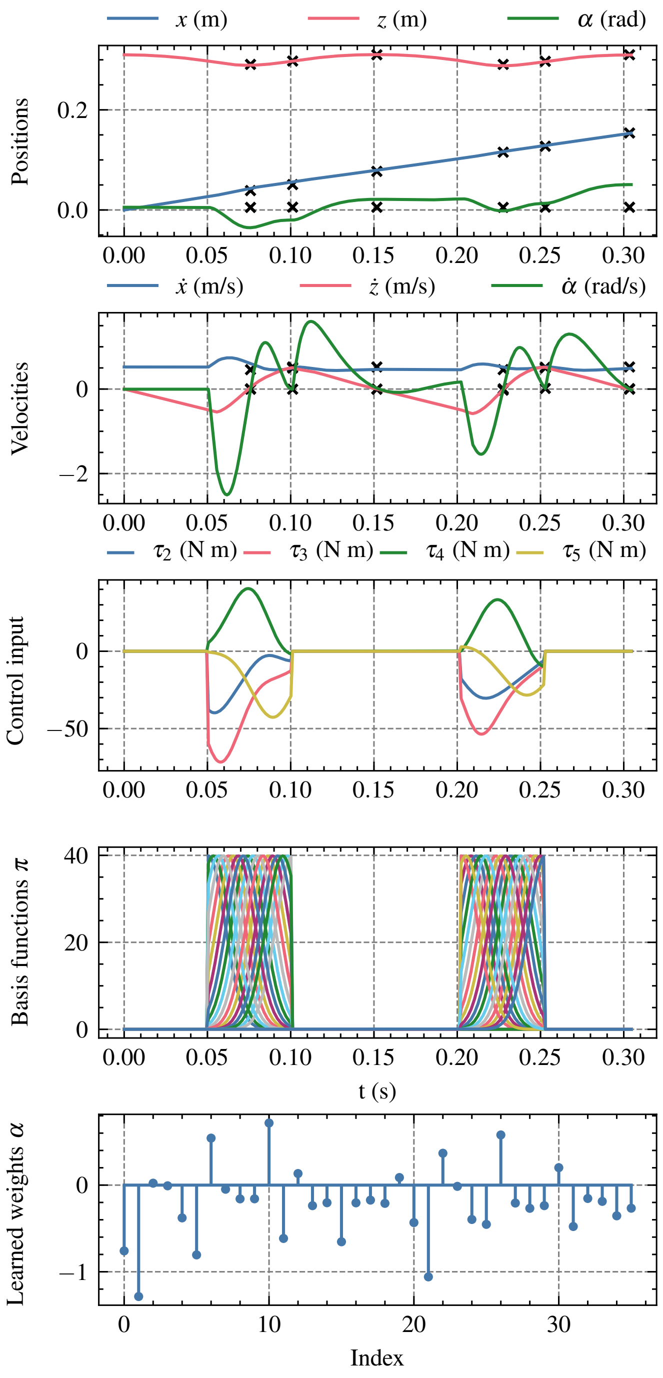


Fig. 17: Evolution of the system state, control input and basis functions after 300 iterations. The bottom panel shows the values of the learned weights. The black crosses indicate the desired state output at a time instance.