TU Delft

Delft University of Technology

Robust Quadrupedal Jumping with Impact-Aware Landing Exploiting Parallel Elasticity

Ding, Jiatao; Atanassov, Vassil; Panichi, Edoardo; Kober, Jens; Santina, Cosimo Della

DOI
10.1109/TRO.2024.3411988

Publication date
2024

Document Version
Final published version

Published in
IEEE Transactions on Robotics

Citation (APA)
Ding, J., Atanassov, V., Panichi, E., Kober, J., & Santina, C. D. (2024). Robust Quadrupedal Jumping with Impact-Aware Landing: Exploiting Parallel Elasticity. *IEEE Transactions on Robotics*, *40*, 3212-3231. https://doi.org/10.1109/TRO.2024.3411988

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Robust Quadrupedal Jumping With Impact-Aware Landing: Exploiting Parallel Elasticity

Jiatao Ding, *Member, IEEE*, Vassil Atanassov, Edoardo Panichi, Jens Kober, *Senior Member, IEEE*, and Cosimo Della Santina, *Senior Member, IEEE*

***Abstract*—Introducing parallel elasticity in the hardware design endows quadrupedal robots with the ability to perform explosive and efficient motions. However, for this kind of articulated soft quadruped, realizing dynamic jumping with robustness against system uncertainties remains a challenging problem. To achieve this, we propose an impact-aware jumping planning and control approach. Specifically, an offline kino-dynamic-type trajectory optimizer is first formulated to achieve compliant 3-D jumping motions, using a novel actuated spring-loaded inverted pendulum (SLIP) model. Then, an optimization-based online landing strategy, including preimpact leg motion modulation and postimpact landing recovery, is designed. The actuated SLIP model, with the capability of explicitly characterizing parallel elasticity, captures the jumping and landing dynamics, making the problem of motion generation/regulation more tractable. Finally, a hybrid torque control consisting of a feedback tracking loop and a feedforward compensation loop is employed for motion control. Experiments demonstrate the ability to accomplish robust 3-D jumping motions with stable landing and recovery. Besides, our approach can be applied to quadrupedal robots with or without additional parallel compliance.**

***Index Terms*—Control, optimization, parallel elasticity, quadrupedal robot.**

I. Introduction

ROBUST jumping in uncertain environments can be realized efficiently by animals through the use of the compliant muscles and tendons system. Transferring this capability to a quadrupedal robot is a long-lasting research topic. Articulated soft designs provide a promising solution to achieve this goal, whereby passive compliance is introduced through the elastic design [1]. In the field of articulated soft quadrupeds with purposefully designed mechanical compliance, the series compliance arrangement, such as [2], [3], and[4] and the parallel arrangement, such as [5], [6], [7], and [8] each come with their own benefits, among which parallel elasticity has the potential to strengthen the joints and improve the energetic performance by providing additional torques. However, the explosive jumping of a soft quadrupedal robot with parallel compliance is still an open problem that needs further investigation.

In terms of jumping with rigid quadrupeds, impressive results [9], [10], [11], [12], [13], and [14] have been achieved by virtue of cutting-edge technologies such as trajectory optimization (TO) and reinforcement learning (RL). However, the above studies, especially the model-based methods, mainly focus on take-off motion generation and control, ignoring the importance of the landing motion. Although several recent works, such as [15], [16], and [17] have paid attention to landing control, they did not realize robust jumping with stable landing and recovery in a unified way. Furthermore, very limited work has been done on quadrupedal jumping by exploiting parallel elasticity. The exception is SpaceBok [8], which is, however, specifically designed to work in a low-gravity environment. Up to now, robust jumps in only such low-gravity environments have been reported [18], [19]. Therefore, no work has so far demonstrated robust jumping of quadrupedal robots with parallel compliance in standard gravity conditions.

This article aims to propose such a control architecture together with experimental validation, both in simulation and on real hardware. More specifically, to successfully jump with the soft quadruped, we notice that the robot should first squat toward the ground to pretension the springs and use the potential energy for an explosive takeoff, followed by a compliant touchdown to ensure a stable landing and recovery. To accomplish this task with robustness against dynamic disturbances, the robot should 1) obey the compliant dynamics, and 2) utilize an impact-aware jumping control strategy. To this end, we first introduce an actuated spring-loaded inverted pendulum (SLIP) model to capture the compliant takeoff and landing dynamics, with the capability of explicitly characterizing the parallel compliance. Based on this reduced-order template model, an offline TO problem is formulated for generating 3-D feasible jumping motion by taking into account kino-dynamic (KD) constraints. Second, an impact-aware landing control strategy is proposed for online landing motion regulation (LMR). Particularly, based on the real takeoff states, real-time quadratic programming (QP) is built to adjust the leg motion in the air. After touchdown, an online TO is then activated for recovery motion regulation.

Experimental results demonstrate that the rigid quadruped Go1 [20] can accomplish versatile 3-D jumps with robustness against dynamic disturbances, including unmodeled center of

Manuscript received 12 November 2023; revised 4 April 2024; accepted 7 May 2024. Date of publication 10 June 2024; date of current version 20 June 2024. This paper was recommended for publication by Associate Editor and Editor Paolo Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported by the EU project under Grant 101016970 NI. *(Corresponding author: Jiatao Ding.)*

Jiatao Ding, Edoardo Panichi, and Jens Kober are with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: J.Ding-2@tudelft.nl; edoardo.panichi99@gmail.com; j.kober@tudelft.nl).

Vassil Atanassov is with the Oxford Robotics Institute, Department of Engineering Science, University of Oxford, OX41EP Oxford, U.K. (e-mail: vassilatanassov@robots.ox.ac.uk).

Cosimo Della Santina is with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands, and also with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany (e-mail: cosimodellasantina@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3411988, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3411988

1941-0468 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

mass (CoM) and angular offsets, external forces, and surface unevenness. Furthermore, this approach is directly applied to the soft E-Go robot, a modified Go1 with the additional design of parallel elastic actuators (PEAs), demonstrating that 1) parallel elasticity encourages more explosive motion. For example, we achieve at least a 20% increase in jumping distance and at least 50% in landing height, and 2) parallel compliance contributes to a lower energy cost in accomplishing a dynamic jumping task.

To summarize, this work contributes to the state of the art in dynamic jumping of quadrupeds as follows.

1) A SLIP-based KD-type[1] TO for 3-D jumping motion generation that can actively exploit parallel elasticity for dynamic jumping without requiring a precomputed reference joint trajectory.
2) An impact-aware LMR. Through online optimization, the preimpact leg motion modulation (LMM) and postimpact landing recovery (LR) endow the robot with robustness against uncertainties.
3) Extensive experiments on quadrupedal jumping with parallel compliance. These validations demonstrate for the first time the controlled robust jumping of a PEA-driven quadruped in standard gravity conditions.

The rest of this article is organized as follows. Section II reviews the related works, followed by an introduction to our methodology in Section III. Section IV and Section V separately present the methodology for SLIP-based jumping motion generation and impact-aware landing control. In Section VI, we introduce the details of motion implementation. Sections VII and VIII separately present simulation and hardware experimental results. Finally, Section IX concludes this article and draws connections with current studies by overall discussions.

II. State of the Art

A. Model-Based Dynamic Jumping With Parallel Compliance

Based on full-body dynamics, the works in [9], [10], [22], and [23] generated the feasible jumping motion, requiring a reference joint trajectory defined in advance. The work in [24] proposed an offline framework without requiring prior knowledge of reference motion and contact schedule, relying on a time-consuming evolutionary search. To improve the computational efficiency, single rigid body (SRB) dynamics, e.g., [11], [15], and [25], and centroidal dynamics, e.g., [12] and [26], could be used, which, however, again require the reference trajectory and assume a fixed stance time. In addition, no parallel elasticity is considered in the above frameworks.

Although recent work [27] has managed to plan natural locomotion using full-body compliant dynamics, no jumping evaluation has been reported. In contrast to the high-order model, SLIP [28] captures the compliant jumping dynamics with very few open parameters. However, the assumption of constant spring stiffness and a fixed rest leg length in the canonical SLIP model [29] limits its application to highly dynamic locomotion, such as high-speed running and jumping. To tackle this issue, researchers introduced the concept called "actuated" SLIP, by changing rest leg length [30], [31], [32], spring constant [33], or force rules [34]. These SLIP variants have been applied in the locomotion of compliant legged robots, such as [35], [36], [37], and [38]. Nevertheless, to the best of the authors' knowledge, there is no application to quadrupedal jumping with parallel elasticity yet.

Two problems must be answered before applying the SLIP model to quadrupedal jumping with parallel compliance. That is, 1) How to model the parallel compliance introduced by the mechanical design? 2) How to guarantee feasibility in motion planning, e.g., satisfying actuation constraints? Regarding point 1), current works estimate the stiffness of the quadruped robot at the system level. Thus, it is hard to highlight the contribution of parallel compliance in executing jumping motions. Regarding point 2), existing SLIP-based motion planners focus on the generation of Cartesian trajectory, including the CoM and leg trajectories, usually ignoring the joint space constraints. As a consequence, an infeasible trajectory could be generated.

To tackle these issues, we first introduce a novel actuated SLIP model, decoupling the system actuation with parallel elasticity. Then, we propose a KD-type TO for jumping planning, simultaneously optimizing the SLIP motion and the joint motions. Particularly, a 3-D TO is built, enabling the robot to accomplish versatile jumping tasks.

B. Stable Landing With Recovery

Aside from the take-off motion, landing with compliant recovery plays a crucial role in robust jumping, which is usually ignored by previous studies. Traditionally, stable landing can be achieved by lower gain feedback control [9], [22], [24], model predictive control (MPC) [10], [39], or full-body control [40]. However, tracking the predefined trajectories itself is not robust enough against severe dynamic uncertainties, such as those caused by modeling mismatch and unknown ground surfaces. Besides, desired tasks such as landing on a certain position cannot be achieved if there are tracking errors at the take-off moment. To tackle these issues, online impact-aware motion modulation and control are required, which, according to the impact status, can be divided into preimpact motion modulation (in the air) and postimpact LR (after touchdown).

To enhance the body reorientation capability in the air, Kolvenbach et al. [18] added a reaction wheel inside the robot, and Kurtz et al. [41] increased the mass of the foot to modulate the mass distribution. Recently, Tang et al. [42] designed a morphable tail on the back of a quadruped. Aside from hardware reformulation, some works, e.g., [43] and [44] modulated the angular momentum through motion control. Nevertheless, the above in-air strategies focus on posture adjustment, ignoring the importance of in-air LMM in reaching desired landing positions [10].

Aside from preimpact motion modulation, postimpact LR motion regulations have also caught a lot of attention. For example, Bingham et al. [45] controlled the robot into a "soft roll" configuration, i.e., a pose that maximizes rolling while also allowing the robot to behave as a damped spring-mass system, during landing. Jeon et al. [15] proposed a novel framework that determines optimal touchdown postures and reaction force profiles for recovering from various falling configurations. Nevertheless, the approaches in [15] and [45] are only applied to vertical falling. Recently, Ye and Karydis [16] evaluated the leg-recovery capability considering body rotation and height variation without reducing the horizontal velocity at the first step. Roscia et al. [17] solved the problem of landing control with aggressive horizontal velocities, which, however, is limited

[1]KD optimization usually refers to the problem formulation obeying centroidal dynamics while fulfilling the full-body kinematic constraints [21]. This work follows the SLIP dynamics while considering joint-level kinematics.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

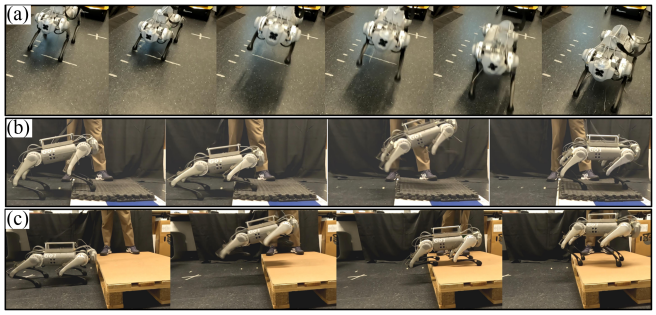

Fig. 1. E-Go robot, i.e., the enhanced Unitree Go1 robot with parallel springs attached on actuated joints, accomplishes multiple jumping tasks: (a) diagonal jumping, i.e., forward 50 cm and leftward 30 cm, in the rigid configuration without spring engaged, (b) jumping off of an unknown pad (5 cm in height) in the compliant configuration with springs engaged, (c) jumping onto a 15 cm-high box in the compliant configuration with springs engaged.

by the assumption of landing on flat ground. Rudin et al. [19] and Qi et al. [46] realized soft landing on uneven terrains using RL. To the best of the authors' knowledge, they only test performance in low-gravity scenarios that enable a long flight phase, and it is still unclear how the performance is in the on-Earth scenario.

In this work, to compensate for the state errors at the take-off moment, we modulate the leg motion in the air, with the aim of reaching the desired landing position. To achieve a stable landing with a quick recovery, we propose a postimpact LR scheme to generate the landing motion online based on the real touch-down status.

III. Methodology Overview

The SLIP-based motion planning and the implementation are illustrated in Fig. 2.

In the motion planning stage, we optimize the jumping motion and landing motion in a unified manner. Particularly, an offline TO is formulated to optimize 3-D jumping motion, considering KD constraints. Based on a novel actuated SLIP model, parallel compliance can be explicitly exploited. After takeoff, a preimpact QP is employed to regulate the leg motion in the air and an online TO is triggered for LR after touchdown.

To realize quadrupedal jumping, we first map the SLIP to the quadrupedal robot. Then, we adopt a hybrid torque ($\boldsymbol{\tau}^r$) control strategy including feedback impedance control ($\boldsymbol{\tau}^{\text{fb}}$) and feedforward torque ($\boldsymbol{\tau}^{\text{ff}}$) compensation for motion tracking. Notably, during the stance and landing phases, a convex MPC strategy is adopted to compute ground reaction forces (GRFs), which are balanced by joint torques ($\boldsymbol{\tau}^{\text{GRF}}$). In addition, when PEA add-ons are engaged, a feedforward compensation for the spring torque (τ^s) is activated.

Notations: In the remaining, matrices and vectors are noted in bold fonts. The superscript $(\cdot)^{\text{T}}$ represents the transpose operation. For the matrix with multiple rows and columns (noted in the bold normal font), the subscript $(\cdot)_{(k)}$ means the kth column and the subscript $(\cdot)_{(j,k)}$ notes the element at the jth row and kth column. In contrast, for the vector (with the size $n \times 1$ or $1 \times n$ that is noted in the bold italic font), $(\cdot)_{(k)}$ refers to the kth element. $\mathbf{I}_n$ is the identity matrix with the size of $n \times n$. Variables accompanied with $(\cdot)^r$ and $(\cdot)^e$ separately denote the reference and estimated values. Besides, variables with the superscript $(\cdot)^{\max}$ and $(\cdot)^{\min}$ separately denote the upper and lower boundaries. In the following, we index the four legs using the pair: front left $\rightarrow$ FL: 1, front right $\rightarrow$ FR: 2, rear left $\rightarrow$ RL: 3, rear right $\rightarrow$ RR: 4.

IV. SLIP-Based 3-D Jumping Motion Generation

A. Dynamics of the Actuated SLIP

The canonical SLIP [29] assumes a lumped mass attached to a mass-less prismatic spring, as illustrated in Fig. 3(a). Consequently, the CoM motion is fully determined by gravity in flight and is regulated by the spring force when the leg touches down. That is, the CoM acceleration is as follows:

$$\begin{aligned} \text{In contact:} \quad & \ddot{\boldsymbol{c}} = \frac{k_s|\boldsymbol{l}_0 - \boldsymbol{l}|\hat{\boldsymbol{l}}}{m} + \boldsymbol{g} \\ \text{In flight:} \quad & \ddot{\boldsymbol{c}} = \boldsymbol{g} \end{aligned} \tag{1}$$

where $\boldsymbol{l} = \boldsymbol{c} - \boldsymbol{p}_f$ ($\boldsymbol{c} \in \mathbb{R}^3$ and $\boldsymbol{p}_f \in \mathbb{R}^3$ separately denote the 3-D CoM and leg position) is the leg vector and $\hat{\boldsymbol{l}}$ is the unit vector along the leg retraction direction, $\ddot{\boldsymbol{c}} \in \mathbb{R}^3$ and $\boldsymbol{g} = [0, 0, -g]^{\text{T}}$ separately denote the 3-D CoM acceleration and gravitational acceleration, with g being the vertical gravitational constant.

Equation (1) tells that, given the initial state, the SLIP motion is totally determined by the spring constant k_s and rest length $|\boldsymbol{l}_0|$, limiting its application to versatile tasks. For example, the initial state should be carefully tuned to accomplish jumps at different speeds. Besides, it is nontrivial to identify the equivalent spring constant k_s of a robotic system.

To alleviate the above limitations, we consider the actuation inputs, which generate an additional driving force when in contact with the ground [see Fig. 3(b)]. As a consequence, we have the actuated SLIP dynamics determined as follows:

$$\begin{aligned} \text{In contact:} \quad & \ddot{\boldsymbol{c}} = \frac{k_s|\boldsymbol{l}_0 - \boldsymbol{l}|\hat{\boldsymbol{l}}}{m} + \boldsymbol{g} + \boldsymbol{u} \\ \text{In flight:} \quad & \ddot{\boldsymbol{c}} = \boldsymbol{g} \end{aligned} \tag{2}$$

where $\boldsymbol{u} \in \mathbb{R}^3$ is the acceleration generated by the actuator.

In this enhanced model, $\boldsymbol{u}$ can be interpreted as the actuation input, which can also be used to capture the variation of the spring constant, rest leg length, or force rules. A detailed explanation from the perspective of Lagrange mechanics is attached in Appendix A. By directly representing it as a virtual actuator we can then gain a more intuitive insight into how it affects the behavior of the system. In (2), k_s is then interpreted as the passive elasticity contributed by the mechanical add-ons, e.g., parallel springs in our E-Go robot. In this sense, k_s is zero if no parallel spring is engaged, which is, however, not allowed in the other SLIP variants.

To apply the novel actuated SLIP model to a quadrupedal system, we should first match the reduced-order model with the full-body model of the E-Go with parallel elasticity. We do this by putting the quadrupedal robot in a homing pose. Using the definition in Fig. 4, the rest leg length equates to the homing height, i.e., $|\boldsymbol{l}_0| = z_0$. Then, we can derive the spring constant k_s for SLIP by considering the PEA design choice, see Appendix B. More details about motion mapping from SLIP to the quadruped are given in Section VI-A.

B. KD-Type TO for 3-D Jumping Motion Generation

We now focus on the jumping motion during the stance phase and the flight phase by assuming that all feet lift off

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

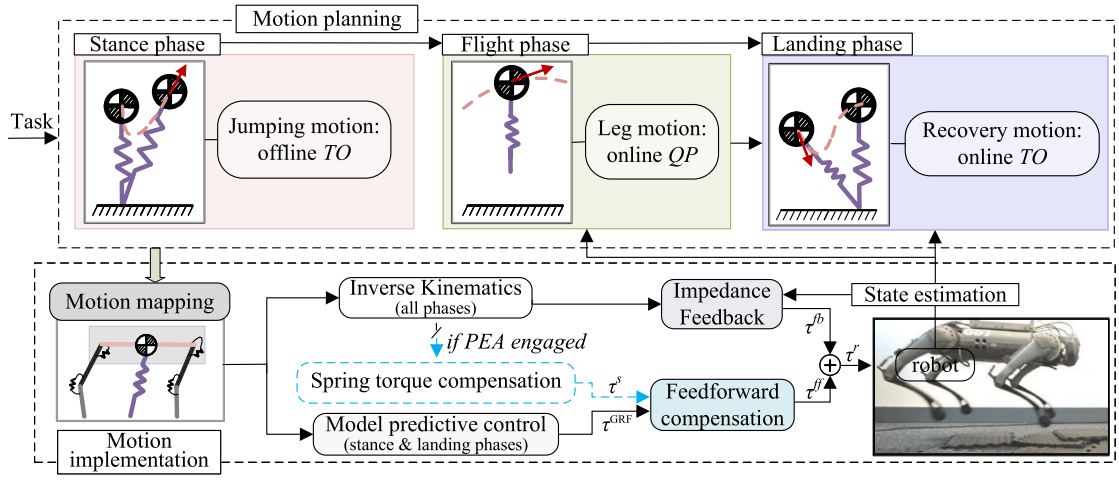

Fig. 2. Motion planning and control approach for 3-D quadrupedal jumping, with the capability of exploiting parallel elasticity. In the "Motion planning" panel, the red arrows mark the movement directions.

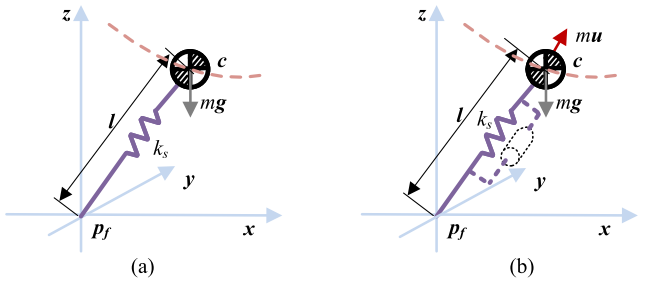

Fig. 3. (a) 3-D canonical SLIP and 3-D actuated SLIP (b). In (b), the input from the actuator (depicted by the dashed cylinder attached to the leg link) is explicitly considered, expressed as $m\boldsymbol{u}$.

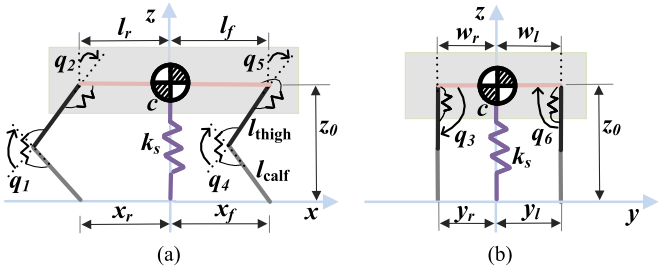

Fig. 4. PEA-driven quadrupedal robot and its SLIP representation. (a) and (b) separately plot the side view and front view when putting the robot in the homing configuration. During the whole jumping process, we assume $x_f = x_r = l_f = l_r$ and $y_l = y_r = w_l = w_r$. $q_1(q_4)$, $q_2(q_5)$, and $q_3(q_6)$ separately denote the joint angle of the calf, thigh, and hip joint.

simultaneously and no body rotation occurs during the stance phase, as depicted in Fig. 5. Assuming N_s knots for the stance phase (the timestep of each knot is t_s) and N_f knots for the flight phase (each knot lasts t_f), we define a TO as follows:

$$\arg \min_{\mathbf{X},\mathbf{u},\mathbf{q},\boldsymbol{t},\xi} J_{\text{cost}} \tag{3a}$$

s.t. **Kinematics constraints** :

$$\mathbf{X}_{(1)} = \boldsymbol{X}_0 \tag{3b}$$

$$\dot{\mathbf{X}}_{(1)} = \dot{\boldsymbol{X}}_0 \tag{3c}$$

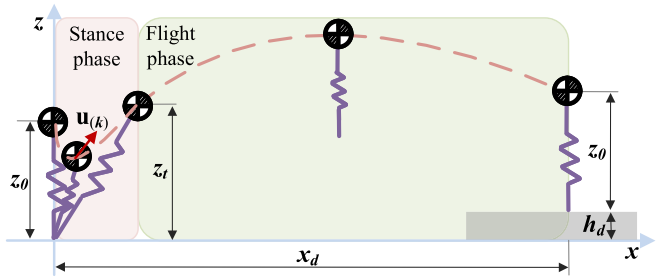

Fig. 5. Example jumping trajectory generated by the KD-type optimization. The red dashed curve represents the CoM trajectory.

$$\mathbf{q}_{(1)} = \boldsymbol{q}_0 \tag{3d}$$

$$(1-\xi)z_t \le \boldsymbol{X}_{z(N_s)} \le (1+\xi)z_t \tag{3e}$$

$$(1-\xi)\lambda^r \le \boldsymbol{X}_{\lambda(N_s+N_f)} \le (1+\xi)\lambda^r, \lambda \in \{x, y\} \tag{3f}$$

$$\boldsymbol{X}_{z(N_s+N_f)} = z^r \tag{3g}$$

$$0 < \xi < \xi^{\max} \tag{3h}$$

$$\boldsymbol{t}^{\min} \le \boldsymbol{t} \le \boldsymbol{t}^{\max} \tag{3i}$$

$$\forall k \in [1, 2, \ldots, N_s + N_f - 1]:$$

$$\mathbf{X}_{(k+1)} = \text{Taylor}(\mathbf{X}_{(k)}, \dot{\mathbf{X}}_{(k)}, \ddot{\mathbf{X}}_{(k)}, t) \tag{3j}$$

$$\dot{\mathbf{X}}_{(k+1)} = \text{Taylor}(\dot{\mathbf{X}}_{(k)}, \ddot{\mathbf{X}}_{(k)}, t) \tag{3k}$$

$$\boldsymbol{q}^{\min} \le \mathbf{q}_{(k)} \le \boldsymbol{q}^{\max} \tag{3l}$$

$$\forall k \in [1, 2, \ldots, N_s]:$$

$$l(\mathbf{X}_{(k)}) \in \text{conv}[\text{leg}] \tag{3m}$$

$$\text{FK}(\mathbf{X}_{(k)}, \mathbf{q}^j_{(k)}) = \boldsymbol{p}^j_f, \ j \in \{1, \ldots, 4\}. \tag{3n}$$

Dynamics constraints :

$$\forall k \in [1, 2, \ldots, N_s]:$$

$$\boldsymbol{\tau}^{\min} \le \boldsymbol{\tau}(\mathbf{X}_{(k)}, \ddot{\mathbf{X}}_{(k)}, \mathbf{q}_{(k)}) \le \boldsymbol{\tau}^{\max} \tag{3o}$$

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

$$\ddot{\mathbf{X}}_{(k)} = \mathbf{F}_s(\mathbf{X}_{(k)})/m + \boldsymbol{g} + \mathbf{u}_{(k)} \tag{3p}$$

$$\boldsymbol{u}_{z(k)} + \boldsymbol{F}_{s,z}(\mathbf{X}_{(k)})/m \geq 0 \tag{3q}$$

$$-\mu \leq \ddot{\boldsymbol{X}}_{\lambda(k)}/(\ddot{\boldsymbol{X}}_{z(k)}+g) \leq \mu,\ \lambda \in \{x, y\} \tag{3r}$$

$$\forall k \in [N_s+1, \ldots, N_s+N_f-1]:$$

$$\ddot{\boldsymbol{X}}_{(k)} = \boldsymbol{g} \tag{3s}$$

where decision variables contain $\mathbf{X}$, $\mathbf{u}$, $\mathbf{q}$, $\boldsymbol{t}$, and ξ. $\mathbf{X} \in \mathbb{R}^{3\times(N_s+N_f)}$ comprises the optimized sagittal ($\boldsymbol{X}_x \in \mathbb{R}^{N_s+N_f}$), lateral ($\boldsymbol{X}_y \in \mathbb{R}^{N_s+N_f}$), and vertical ($\boldsymbol{X}_z \in \mathbb{R}^{N_s+N_f}$) positions at all knots, $\mathbf{u} \in \mathbb{R}^{3\times N_s}$ denotes the optimized sagittal ($\boldsymbol{u}_x \in \mathbb{R}^{N_s}$), lateral ($\boldsymbol{u}_y \in \mathbb{R}^{N_s}$), and vertical ($\boldsymbol{u}_z \in \mathbb{R}^{N_s}$) accelerations (i.e., control inputs) during the stance phase, $\mathbf{q} \in \mathbb{R}^{12\times N_s}$ consists of optimized joint angles of four legs ($\mathbf{q}^{\text{FL}}$, $\mathbf{q}^{\text{FR}}$, $\mathbf{q}^{\text{RL}}$, and $\mathbf{q}^{\text{RR}} \in \mathbb{R}^{3\times N_s}$), $\boldsymbol{t} = [t_s, t_f]^{\text{T}} \in \mathbb{R}^2$ contains the step sizes for stance and flight phases, $\xi \in \mathbb{R}$ is the slack variable. Taylor$(\cdot)$ denotes the Taylor expansion.

1) Cost Function: The cost function in (3a) is defined as follows:

$$J_{\text{cost}} = J_{\text{stance}} + J_{\text{takeoff}} + J_{\text{flight}} + J_{\text{land}} + J_t + J_q + J_\xi \tag{4a}$$

with

$$J_{\text{stance}} = \sum_{k=1}^{N_s} w_u \|\mathbf{u}_{(k)}\|^2 + \sum_{k=1}^{N_s-1} w_a \|\ddot{\mathbf{X}}_{(k+1)} - \ddot{\mathbf{X}}_{(k)}\|^2 \tag{4b}$$

$$J_{\text{takeoff}} = -w_{vx}\|\dot{\boldsymbol{X}}_{x(N_s)}\|^2 - w_{vy}\|\dot{\boldsymbol{X}}_{y(N_s)}\|^2 - w_{vz}\|\dot{\boldsymbol{X}}_{z(N_s)}\|^2 \tag{4c}$$

$$J_{\text{flight}} = \sum_{k=N_s+1}^{N_s+N_f} w_f \|\boldsymbol{X}_{z(k)} - z_t\|^2 \tag{4d}$$

$$J_{\text{land}} = w_x\|\boldsymbol{X}_{x(Ns+Nf)} - x^r\|^2 + w_y\|\boldsymbol{X}_{y(Ns+Nf)} - y^r\|^2 \tag{4e}$$

$$J_t = w_t\|\boldsymbol{t} - \boldsymbol{t}^r\|^2 \tag{4f}$$

$$J_q = \sum_{k=1}^{N_s} w_q \|\mathbf{q}_{(k)} - \boldsymbol{q}_0\|^2 \tag{4g}$$

$$J_\xi = w_s\|\xi\|^2 \tag{4h}$$

where J_{stance} penalizes control inputs and the acceleration increments to achieve smooth behavior, J_{takeoff} rewards high shooting velocities to reach desired landing positions, J_{flight} penalizes deviations from the nominal takeoff height z_t during the flight phase to avoid unrealistic height, J_{land} minimizes the tracking error of the desired landing position, i.e., $[x^r, y^r]$, J_t penalizes the deviation from the reference time steps ($\boldsymbol{t}^r = [t_s^r, t_f^r]^{\text{T}} \in \mathbb{R}^2$ are tuned by hand), J_q penalizes the joint angle variations from the initial angles ($\boldsymbol{q}_0 \in \mathbb{R}^{12}$ is the initial joint angles in the homing state), w_u, w_a, w_{vx}, w_{vy}, w_{vz}, w_f, w_x, w_y, w_t, w_q, and w_s are the positive weights.

2) KD Constraints: The following kinematic constraints (3b)–(3n) and dynamic constraints (3o)–(3s) are taken into consideration in jump motion generation.

Kinematics constraints: Equations (3b)–(3d) define the initial jumping state, i.e., the homing pose, where $\boldsymbol{q}_0$ is the initial joint angle computed by inverse kinematics. As plotted in Fig. 4(a), we have the initial state with zero velocity

$$\boldsymbol{X}_0 = [0, 0, z_0]^{\text{T}},\ \dot{\boldsymbol{X}}_0 = [0, 0, 0]^{\text{T}} \tag{5}$$

with z_0 being the homing height.

Equation (3e) limits the shooting height at the take-off moment. By using inequality constraints, we do not need to tune the shooting height by hand for different jumping tasks. In practice, we found that in order to jump longer and higher, the robot should shoot at a big height with a large shooting velocity as much as possible. Thus, we can give an initial guess of the nominal shooting height as follows:

$$z_t = \sqrt{(l_{\text{thigh}} + l_{\text{calf}})^2 - l_f^2} \tag{6}$$

where l_{thigh}, l_{calf}, and l_f separately denote the thigh-link length, calf-link length, and front thigh offset, as depicted in Fig. 4(a).

Equations (3f) and (3g) define the terminal conditions at the end of the flight phase (i.e., the touch-down moment), with λ^r in (3f) ($\lambda \in \{x, y\}$) being the target jumping distance. As plotted by Fig. 5, given a desired landing height (h_d), z^r in (3g) is determined as follows:

$$z^r = z_0 + h_d. \tag{7}$$

To enhance the solvability, we use soft constraints in (3e) and (3f) by introducing a slack variable ξ, which is clamped by an inequality constraint expressed in (3h). In (3h), the upper boundary $\xi^{\max}$ is computed to obey the height limits, e.g.,

$$\xi^{\max} = \sqrt{(l_{\text{thigh}} + l_{\text{calf}})^2}/z_t - 1. \tag{8}$$

The soft constraints in (3f) allow in a minor error between the generated landing position (at the (N_s+N_f)th node) and the desired position, which can be compensated by modulating leg motions in the flight phase. Note that we do not introduce ξ in (3g) because we expect the robot to land in the homing pose at the end of the flight phase.

Equation (3i) limits the step size, with $\boldsymbol{t}^{\min} \in \mathbb{R}^2$ and $\boldsymbol{t}^{\max} \in \mathbb{R}^2$ being the minimal and maximal values, respectively.

Equations (3j) and (3k) describe the state transition between neighboring knots. In this work, we adopt the Euler integration to guarantee the continuity, using the respective step size t, i.e., t_s ($\boldsymbol{t}(1)$) for the stance phase and t_f ($\boldsymbol{t}(2)$) for the flight phase, and the corresponding acceleration (computed using (3p) during the stance phase and by (3s) during the flight phase). Equations (3j) and (3k) are specialized by Taylor expansion as follows:

$$\begin{aligned}\mathbf{X}_{(k+1)} &= \mathbf{X}_{(k)} + \dot{\mathbf{X}}_{(k)}t + 0.5\ddot{\mathbf{X}}_{(k)}t^2 \\ \dot{\mathbf{X}}_{(k+1)} &= \dot{\mathbf{X}}_{(k)} + \ddot{\mathbf{X}}_{(k)}t.\end{aligned} \tag{9}$$

Equation (3m) limits the leg extension and retraction to obey the kinematic reachability. To be specific, we have the following:

$$l^{\min} \leq l(\mathbf{X}_{(k)}) = \sqrt{\boldsymbol{X}_{x(k)}^2 + \boldsymbol{X}_{y(k)}^2 + \boldsymbol{X}_{z(k)}^2} \leq l^{\max} \tag{10}$$

where $l^{\min}$ and $l^{\max}$ separately denote the minimal and maximal leg length, determined by the joint limits.

Equation (3l) restricts the joint angles, where the upper boundary ($\boldsymbol{q}^{\max} \in \mathbb{R}^{12}$) and the lower boundary ($\boldsymbol{q}^{\min} \in \mathbb{R}^{12}$) are determined by the hardware design.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

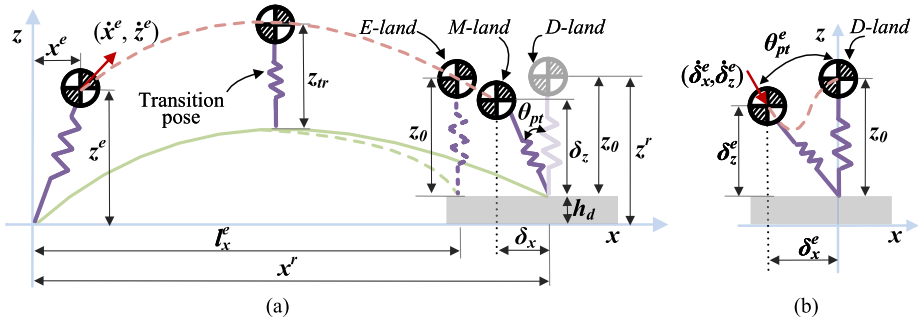

Fig. 6. Impact-aware LMR (taking the sagittal motion for an example). (a) LMM in the air. The red dashed curve plots the CoM trajectory. The green dashed and solid curves separately plot the estimated leg trajectory before LMM and the modulated leg trajectory after LMM. (b) SLIP-based LR after touchdown.

Equation (3n) restricts the leg movements during the stance phase to avoid the slippage of support feet. That is, the constraints in (3n) are given as follows:

$$\mathrm{FK}(\mathbf{X}_{(k)}, \mathbf{q}^j_{(k)}) = \boldsymbol{p}^j_f \tag{11}$$

where $\boldsymbol{p}^j_f$ is the jth leg position during the stance phase (i.e., $[x_f, y_l, 0]^{\mathrm{T}}$, $[x_f, -y_l, 0]^{\mathrm{T}}$, $[-x_r, y_l, 0]^{\mathrm{T}}$, and $[-x_r, -y_l, 0]^{\mathrm{T}}$, as illustrated in Fig. 4). $\mathrm{FK}(\cdot)$ denotes the forward kinematics operation [47].

Dynamics constraints: Equation (3o) restrict the joint torques, where $\boldsymbol{\tau}(\cdot)$ in (3o) denotes the operation for computing the commanded torque, see Appendix C. Since all the legs are presumed to be massless, we do not limit the torque during the flight phase.

Equation (3p) computes the CoM acceleration during the stance phase, where the spring forces ($\mathbf{F}_s(\mathbf{X}) \in \mathbb{R}^{3\times N_s}$) are computed by the first row in (2). Specifically

$$\mathbf{F}_s(\mathbf{X}_{(k)}) = \begin{bmatrix} \boldsymbol{F}_{s,x}(\mathbf{X}_{(k)}) \\ \boldsymbol{F}_{s,y}(\mathbf{X}_{(k)}) \\ \boldsymbol{F}_{s,z}(\mathbf{X}_{(k)}) \end{bmatrix} = \frac{k_s(l(\mathbf{X}_{(k)}) - z_0)}{l(\mathbf{X}_{(k)})} \begin{bmatrix} \boldsymbol{X}_{x(k)} \\ \boldsymbol{X}_{y(k)} \\ \boldsymbol{X}_{z(k)} \end{bmatrix} \tag{12}$$

where $l(\mathbf{X}_{(k)})$ is computed by (10), $\boldsymbol{F}_{s,x}(\mathbf{X}) \in \mathbb{R}^{1\times N_s}$, $\boldsymbol{F}_{s,y}(\mathbf{X}) \in \mathbb{R}^{1\times N_s}$, and $\boldsymbol{F}_{s,z}(\mathbf{X}) \in \mathbb{R}^{1\times N_s}$ separately represent the sagittal, lateral, and vertical components of the spring forces during the stance phase.

Equation (3q) restricts the vertical acceleration to avoid a fall.

Equation (3r) prevents slippages where μ is the friction coefficient.

Equation (3s) computes the CoM acceleration in flight.

Using the above formulation, we can generate the 3-D optimal jumping motion for a given jumping task. One example trajectory is plotted by Fig. 5.

Remark 1: Differing from the jumping motion planners in [9], [10], [11], [12], [22], and[23], our TO formulation does not need a reference joint trajectory for the whole jumping motion. Furthermore, unlike many SLIP-based motion planners, e.g., [35], [48], and [49], we do not optimize the touchdown motion, which can be characterized by the touch-down angle [see "θ_{pt}" in Fig. 6(a)], in this TO formulation. Instead, we resort to an impact-aware LMR strategy to achieve a stable landing pose.

V. Impact-Aware LMR

Triggered by the landing event, LMR is decomposed into a preimpact LMM and a postimpact LR. Particularly, the LMM adjusts the leg motion in the air according to the real take-off state. Then, based on the landing state, the LR generates the recovery motion for a stable landing with compliance.

A. Preimpact LMM

Due to dynamic disturbances, e.g., modeling mismatch, the reference shooting state at the take-off moment may not be well tracked, resulting in an undesired landing location. To correct this, we modulate the leg configuration before touchdown, assuming that the leg movement does not change the body orientation due to the lumped mass assumption.

Before clarifying our design choice, we define three landing states, which are illustrated in Fig. 6(a). That is

Desired landing state (D-land): It consists of the desired CoM ($[x^r, y^r, z^r]^{\mathrm{T}}$) and leg ($[x^r, y^r, h_d]^{\mathrm{T}}$) position when landing.

Estimated landing state (E-land): It consists of the estimated leg and CoM position when landing, assuming that the robot touches down in the homing pose at the desired landing height. Here the estimated flight time (δ^e_t) and flight distance ($[l^e_x, l^e_y]^{\mathrm{T}}$), which are computed based on the real shooting states [consisting of shooting position ($[x^e, y^e, z^e]^{\mathrm{T}}$) and velocity ($[\dot{x}^e, \dot{y}^e, \dot{z}^e]^{\mathrm{T}}$)], are used to characterize the *E-land* state.

Modulated landing state (M-land): It consists of the modulated leg position and CoM position for landing. Specifically, the horizontal distance offsets between the leg and CoM position that are denoted as δ_x, δ_y (by default, $\delta_x = \delta_y = 0$), and landing height δ_z (by default, $\delta_z = z_0$) are used to describe the modulated landing configuration. $[\delta_x, \delta_y, \delta_z]^{\mathrm{T}}$ are computed online so as to reach the desired landing position.

Considering the modulated landing height δ_z can be computed using the modulated flight time (denoted as δ_t), we here choose δ_t, δ_x, and δ_y as decision variables. We then modulate the leg motion by solving the following QP

$$\arg\min_{\delta_t,\delta_x,\delta_y} J = \alpha_t(\delta_t - \delta^r_t)^2 + \beta_x(\delta_x)^2 + \gamma_x(x_{\text{land}} - x^r)^2 + \beta_y(\delta_y)^2 + \gamma_y(y_{\text{land}} - y^r)^2 \tag{13a}$$

$$\text{s.t.} \quad \delta^{\min}_\zeta \le \delta_\zeta \le \delta^{\max}_\zeta, \qquad \zeta \in \{t, x, y\} \tag{13b}$$

$$\lambda_{\text{land}} = \dot{\lambda}^e \delta_t + \lambda^e + \delta_\lambda, \quad \lambda \in \{x, y\} \tag{13c}$$

where δ^r_t is the reference flight time, x_{land} and y_{land} are the modulated sagittal and lateral landing position, $[\delta^{\min}_t, \delta^{\max}_t]^{\mathrm{T}}$, $[\delta^{\min}_x, \delta^{\max}_x]^{\mathrm{T}}$, and $[\delta^{\min}_y, \delta^{\max}_y]^{\mathrm{T}}$ separately denote the boundary values of δ_t, δ_x, and δ_y. $\alpha_t, \beta_x, \beta_y, \gamma_x$, and γ_y are positive penalty coefficients.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

1) Objective Function: The objective function defined in (13a) enables the robot tracking δ_t^r, x^r, and y^r. In particular, δ_t^r is chosen in such a way that the robot would jump onto the desired distance, i.e.,

$$\delta_t^r = \max\{\delta_t^{xe}, \delta_t^{ye}, \delta_t^{ze}\} \tag{14}$$

with $[\delta_t^{xe}, \delta_t^{ye}, \delta_t^{ze}]^{\mathrm{T}}$ being the online-estimated flight time for reaching the desired 3-D landing position, which are determined by solving the following:

$$\begin{aligned} \delta_t^{xe}\dot{x}^e + x^e &= x^r \\ \delta_t^{ye}\dot{y}^e + x^e &= y^r \\ -\frac{g}{2}(\delta_t^{ze})^2 + \dot{z}^e\delta_t^{ze} + z^e &= h_d + z^0. \end{aligned} \tag{15}$$

In the third row of (15), δ_t^{ze} is chosen to be the larger value between the two solutions of the second-order equation.

2) Feasibility Constraints: Equation (13b) limits the variation of δ_t, δ_x, and δ_y. Specifically, $\delta_t^{\min}$ is achieved when the robot lands with the leg fully stretched (i.e., with $z^{\max} = l_{\text{thigh}} + l_{\text{calf}}$) and $\delta_t^{\max}$ is achieved when the robot lands in the minimal height (i.e., with $z^{\min}$ in height). That is, we compute $\delta_t^{\min}$ and $\delta_t^{\max}$ by solving the following:

$$\begin{aligned} -\frac{g}{2}(\delta_t^{\min})^2 + \dot{z}^e\delta_t^{\min} + z^e &= h_d + z^{\max} \\ -\frac{g}{2}(\delta_t^{\max})^2 + \dot{z}^e\delta_t^{\max} + z^e &= h_d + z^{\min} \end{aligned} \tag{16}$$

where $z^{\min}$ is tuned by hand obeying the kinematics limits.

The hyperparameters, i.e., $\delta_x^{\min}$, $\delta_x^{\max}$, $\delta_y^{\min}$, and $\delta_y^{\max}$ in (13b) are determined by the task. For example, when jumping forward, we have $\delta_x^{\max} > \delta_x^{\min} = 0$ such that the robot would stop in the landing position without falling forward.

Equation (13c) computes the modulated sagittal jumping distance x_{land} and lateral jumping distance y_{land} by the fact that there is no horizontal force imposed on the robot in flight.

Since the above QP is solved very fast, the preimpact LMM can be accomplished in real-time. After solving δ_t, the modulated height δ_z is determined as follows:

$$\delta_z = \max\left\{\min\{-\frac{g}{2}(\delta_t)^2 + \dot{z}^e\delta_t + z^e - h_d, \sqrt{z_0^2 - \delta_x^2 - \delta_y^2}\}, z^{\min}\right\}. \tag{17}$$

Remark 2: Using LMM, the robot would touch down following the *M-land* configuration. Considering that the real shooting state is prone to deviating from the reference one, *M-land* usually does not coincide with *D-land*. That is, the SLIP would land with a nonzero touch-down angle. Stopping the robot in *M-land* configuration could stabilize the robot but may result in an undesired pose. For example, when jumping forward at a long distance, the robot would stop with its body leaning backward, resulting in a negative pitch angle. We will discuss it in detail in Section VII-C1.

B. Postimpact LR

To reduce the landing impact and quickly return to the homing state (i.e., *D-land* configuration), we propose a postimpact LR scheme after detecting the landing event. Currently, we assume that the robot can stop without making further steps after touch-down. Besides, differing from the previous work, e.g., [17], we do not assume that the robot lands on flat ground.

Given the touch-down state, i.e., postimpact CoM position ($[\delta_x^e, \delta_y^e, \delta_z^e]^{\mathrm{T}}$) and velocity ($[\dot{\delta}_x^e, \dot{\delta}_y^e, \dot{\delta}_z^e]^{\mathrm{T}}$) relative to the landing position [see Fig. 6(b)], we solve a light-weighed TO for real-time LR.

Assuming N_l knots during the landing phase with each knot lasting time t_l, we define the following TO problem:

$$\arg\min_{\mathbf{X}^l, \mathbf{u}^l, t_l} \; J_{\text{cost}}^l \tag{18a}$$

$$\text{s.t.} \quad \textbf{Kinematics constraints}: \tag{18b}$$

$$\mathbf{X}_{(1)}^l = \boldsymbol{X}_0^e \tag{18c}$$

$$\dot{\mathbf{X}}_{(1)}^l = \dot{\boldsymbol{X}}_0^e \tag{18d}$$

$$\lambda_l^{\min} \le \boldsymbol{X}_{\lambda(N_l)}^l \le \lambda_l^{\max}, \quad \lambda \in \{x, y, z\} \tag{18e}$$

$$t_l^{\min} \le t_l \le t_l^{\max} \tag{18f}$$

$$\forall k \in [1, 2, \ldots, N_l - 1]:$$

$$\text{State transition in (9)} \tag{18g}$$

$$l^{\min} \le l(\mathbf{X}_{(k)}) \le l^{\max} \tag{18h}$$

$$\textbf{Dynamics constraints}: \tag{18i}$$

$$\forall k \in [1, 2, \ldots, N_l]: \text{SLIP dynamics in (3p)} \tag{18j}$$

where $\mathbf{X}^l \in \mathbb{R}^{3\times N_l}$ comprises the optimized sagittal ($\boldsymbol{X}_x^l \in \mathbb{R}^{N_l}$), lateral ($\boldsymbol{X}_y^l \in \mathbb{R}^{N_l}$), and vertical ($\boldsymbol{X}_z^l \in \mathbb{R}^{N_l}$) positions at all knots, $\mathbf{u}^l \in \mathbb{R}^{3\times N_l}$ comprises the sagittal ($\boldsymbol{u}_x^l \in \mathbb{R}^{N_l}$), lateral ($\boldsymbol{u}_y^l \in \mathbb{R}^{N_l}$), and vertical ($\boldsymbol{u}_z^l \in \mathbb{R}^{N_l}$) component of control inputs.

1) Cost Function: The cost function in (18a) is defined as follows:

$$J_{\text{cost}}^l = J_{\text{land}}^l + J_{\text{track}}^l + J_{\text{stop}}^l + J_t^l \tag{19a}$$

with

$$J_{\text{land}}^l = \sum_{k=1}^{N_l} w_u^l \|\mathbf{u}_{(k)}^l\|^2 + \sum_{k=1}^{N_l-1} w_a^l \|\ddot{\mathbf{X}}_{(k+1)}^l - \ddot{\mathbf{X}}_{(k)}^l\|^2 \tag{19b}$$

$$J_{\text{track}}^l = w_p^l \|\mathbf{X}_{(N_l)} - \mathbf{X}_{(N_l)}^r\|^2 \tag{19c}$$

$$J_{\text{stop}}^l = w_v^l \|\dot{\mathbf{X}}_{(N_l)}\|^2 \tag{19d}$$

$$J_t^l = w_t^l \|t_l - t_l^r\|^2 \tag{19e}$$

where J_{cost}^l penalizes control inputs and the acceleration variations, J_{track}^l penalizes the tracking error of the homing position ($\mathbf{X}_{(N_l)}^r = [0, 0, z_0]^{\mathrm{T}}$) at the end of the landing phase, J_{stop}^l penalizes the final CoM velocity, J_t^l penalizes the deviation of step size, w_u^l, w_a^l, w_p^l, w_v^l, and w_t^l are positive penalty weights.

2) Feasibility Constraints: Equations (18c) and (18d) define the initial conditions for recovery motion, which are given by the estimated touch-down state, i.e.,

$$\boldsymbol{X}_0^e = [\delta_x^e, \delta_y^e, \delta_z^e]^{\mathrm{T}}, \quad \dot{\boldsymbol{X}}_0^e = [\dot{\delta}_x^e, \dot{\delta}_y^e, \dot{\delta}_z^e]^{\mathrm{T}}. \tag{20}$$

Equation (18e) defines the terminal conditions at the end of the landing phase. To improve the solvability, we allow the final CoM to move within a small range, clamped by the boundaries

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

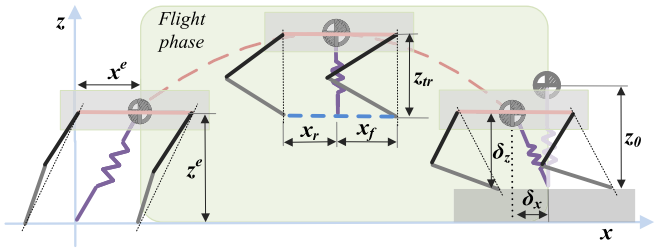

Fig. 7. Motion mapping from the SLIP model to the quadrupedal robot, taking the sagittal motion as an example.

$\lambda_l^{\min}$ and $\lambda_l^{\max}$ ($\lambda \in \{x, y, z\}$).[2] Besides, we do not impose strict constraints on the final CoM velocity.

Equation (18f) guarantees a feasible step size, with $t_l^{\min}$ and $t_l^{\max}$ being the minimal and maximal values, respectively.

Equation (18g) describes the state transition between neighboring knots, following the rules defined in (9).

Equation (18h) constrains the leg length to obey the kinematic limit, using the same rule in (10).

Equation (18j) describes the SLIP dynamics, sharing the same dynamic property with the stance phase.

Remark 3: Unlike the KD-type TO defined in Section IV-B, we do not optimize the joint-level motion, allowing an aggressive recovery motion. Besides, friction constraints are not obeyed anymore. This simplification makes it possible to achieve a time-efficient solution. The usage of the online LR scheme would help to maintain balance in uncertain environments and achieve a quick recovery behavior with compliance. Note that the recovery trajectory is computed in the local support coordinate, alleviating the need for an accurate estimation of the global landing position.

If not differently specified, LMR in the following parts includes LMM (in Section V-A) and LR (in Section V-B).

VI. Motion Implementation

This section clarifies implementation details for executing the jumping and landing motions. First, we present the motion mapping strategy to achieve executable quadrupedal motions. Then, we briefly introduce the low-level control strategy to complete the system.

A. Motion Mapping: From SLIP to Quadruped

Assuming massless legs, the CoM coincides with the trunk center. Considering the optimal CoM trajectory is generated at a low frequency, we use linear interpolation to generate a reference trajectory for high-frequency low-level control.

Since there is no leg movement when getting in contact with the ground, the leg positions of the quadrupedal robot during the stance and landing phases are directly determined by the SLIP leg position, using the rules revealed in Fig. 4. Therefore, we focus on the leg motions during the flight phase. To this end, we split the leg movement in the air (in the body frame) into two segments, following the principles revealed in Fig. 7. Specifically, before reaching the peak height (the peak time is estimated by $t_p = \dot{z}^e/g$), we drive the leg to move from take-off position ($[-x^e, -y^e, -z^e]^{\mathrm{T}}$) to the transitional position ($[0, 0, -z_{tr}]^{\mathrm{T}}$). Then, the leg moves toward *E-land* configuration, i.e., $[\delta_x, \delta_y, -\delta_z]^{\mathrm{T}}$ within a quarter of flight phase (the time period was tuned by experience to make sure the *E-landing* configuration could be reached before touchdown). Here, we use the logistic function, i.e., $1/(1+e^{a(t-b)})$, to achieve a smooth motion passing through the above key points. After obtaining the SLIP leg motion, we obtain the quadrupedal leg position by assuming the rear leg, front leg, and SLIP leg coexist in a straight line (marked by the blue dashed line Fig. 7), aligning with the horizontal surface. In particular, we have $x_r = x_f$ and $y_r = y_l$.

Remark 4: In Fig. 7, we set the transitional leg position to fall below the body center with z_{tr} in vertical height. By default, $z_{\text{tr}} = z_0$. In real applications, z_{tr} can be changed for versatile tasks, e.g., jumping above an obstacle. More discussions can be found in Section VII-C3.

Note that when integrating LMM, the measured body inclinations in the air are incorporated in inverse kinematics. Particularly, after reaching the peak height, the reference rotation angle is generated using the logistics interpolation, starting from zeros to the measured values. After landing, when LR is engaged, a smooth rotation trajectory is generated by logistics interpolation such that the robot returns to zero inclination. In this work, we do not estimate the inclination of the contact surface. Thus, there could still be body inclinations after LR when landing on uneven surfaces.

B. Low-Level Torque Control

This section briefly introduces the torque control strategy.

1) MPC-Based GRF Compensation: We introduce the following MPC formulation for GRF generation:

$$\arg\min_{\mathbf{\Gamma},\mathbf{F}} \sum_{k=1}^{N_h} \|\mathbf{S}_{(k)} - \mathbf{S}^r_{(k)}\|^2_{\mathbf{Q}} + \|\mathbf{\Gamma}_{(k)}\|^2_{\mathbf{R}} \tag{21a}$$

$$\text{s.t.} \quad \mathbf{S}_{(k)} = \mathbf{A}\mathbf{S}_{(k-1)} + \mathbf{B}\mathbf{\Gamma}_{(k)} \,(\text{with}\, \mathbf{S}_{(0)} = \boldsymbol{S}^e) \tag{21b}$$

$$\mathbf{\Gamma}_{(k)} = \begin{bmatrix} \mathbf{I}_{3\times3} & \cdots & \mathbf{I}_{3\times3} \\ [\mathbf{p}^1_{(k)} - \mathbf{c}_{(k)}]_\times & \cdots & [\mathbf{p}^4_{(k)} - \mathbf{c}_{(k)}]_\times \end{bmatrix} \mathbf{F}_{(k)} + \begin{bmatrix} \boldsymbol{g} \\ \mathbf{0}_{3\times1} \end{bmatrix} \tag{21c}$$

$$d(\mathbf{p}^j_{(k)})\mathbf{F}^j_{(k)} = \mathbf{0} \;(d(\mathbf{p}^j_{(k)}) \geq 0),\, j \in \{1, \ldots, 4\} \tag{21d}$$

$$0 \leq \mathbf{F}^j_{(3,k)} \leq F_z^{\max} \tag{21e}$$

$$-\mu\mathbf{F}^j_{(3,k)} \leq \mathbf{F}^j_{(i,k)} \leq \mu\mathbf{F}^j_{(3,k)},, i \in \{1, 2\} \tag{21f}$$

where state $\mathbf{S} = [\boldsymbol{c}_x, \boldsymbol{c}_y, \boldsymbol{c}_z, \boldsymbol{\theta}_r, \boldsymbol{\theta}_p, \boldsymbol{\theta}_y, \dot{\boldsymbol{c}}_x, \dot{\boldsymbol{c}}_y, \dot{\boldsymbol{c}}_z, \dot{\boldsymbol{\theta}}_r, \dot{\boldsymbol{\theta}}_p, \dot{\boldsymbol{\theta}}_y]^{\mathrm{T}} \in \mathbb{R}^{12\times N_h}$ consists of 3-D CoM position, CoM velocity, body angles, and angular velocity, $\boldsymbol{S}^e \in \mathbb{R}^{12}$ is the measured state, control inputs include the linear and angular accelerations, i.e., $\mathbf{\Gamma} = [\ddot{\boldsymbol{c}}_x, \ddot{\boldsymbol{c}}_y, \ddot{\boldsymbol{c}}_z, \ddot{\boldsymbol{\theta}}_r, \ddot{\boldsymbol{\theta}}_p, \ddot{\boldsymbol{\theta}}_y]^{\mathrm{T}} \in \mathbb{R}^{6\times N_h}$, and GRFs, i.e., $\mathbf{F} \in \mathbb{R}^{12\times N_h}$. $\mathbf{F}^j \in \mathbb{R}^{3\times N_h} (j \in \{1, \ldots, 4\})$ denotes the GRF of each leg. $\mathbf{p}^j_{(k)} \in \mathbb{R}^3$ and $\mathbf{c}_{(k)} \in \mathbb{R}^3$ separately denote the jth leg position and the 3-D CoM position at the kth step.

[2] In this case, the final state generated by the TO may not coincide with the homing pose. In real applications, the robot is driven to the homing pose by tracking a consecutive segment of the reference trajectory that is generated by a polynomial interpretation. We will demonstrate it in Section VII-C1.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

Objective function: The objective function in (21a) penalizes the tracking error with minimal control efforts. The reference CoM position and velocity in $\mathbf{S}^r$ are generated by the strategy in Section IV-B and Section V. The reference body angles and angular velocities are zeros during the stance phase. In contrast, during the landing phase, the reference body angles are nonzeros, as described in Section VI-A. N_h is the prediction horizon length. $\mathbf{Q} \in \mathbb{R}^{12\times12}$ and $\mathbf{R} \in \mathbb{R}^{12\times12}$ are the diagonal weight matrices.

Feasibility constraints: The state transition is processed in (21b), where $\mathbf{A}$ and $\mathbf{B}$ are computed offline following the Taylor expansion.

Equation (21c) encapsulates the SRB dynamics where $[\cdot]_\times$ denotes cross-product operation. For the first step $k = 1$, $\mathbf{\Gamma}_{(1)}$ is computed through the forward kinematics. For the remaining steps within the prediction window, it is precomputed based on the desired CoM position relative to the foot position.

The contact complementary conditions in (21d) ensure that GRFs are only assigned to the feet that are in contact with the surface, where $\mathbf{F}^j_{(k)} \in \mathbb{R}^3$ is the 3-D GRF exerted on the jth foot at the future kth step, and $d(\mathbf{p}^j_{(k)})$ is the non-negative distance metric between the ground and the jth foot. Particularly, $d(\mathbf{p}^j_{(k)}) = 0$ when the robot touches the ground.

Equations (21e) and (21f) ensure that the GRFs stay within the friction cone. Besides, we also restrict the maximum normal component. $\mathbf{F}^j_{(1,k)} \in \mathbb{R}$, $\mathbf{F}^j_{(2,k)} \in \mathbb{R}$, and $\mathbf{F}^j_{(3,k)} \in \mathbb{R}$ separately denote the sagittal, lateral, and vertical GRF components on the jth foot. $F_z^{\max}$ is the maximal vertical force tuned by hand.

After solving the GRF, the feedforward torques ($\boldsymbol{\tau}^{\text{GRF}}$) for GRF compensation are computed, following (35).

2) Spring Force Compensation: The additional spring torques in the 12 actuated joints introduced by parallel springs need to be compensated. Given the reference joint angles ($\boldsymbol{q}^r \in \mathbb{R}^{12}$), we model the spring torques using the softplus function to achieve a smoother behavior. That is

$$\boldsymbol{\tau}^s = \begin{cases} \log(\mathbf{1}_{12} + \mathrm{e}^{\mathbf{k}_q(\boldsymbol{q}^r - \boldsymbol{q}^0)}) & \text{spring engaged} \\ \mathbf{0}_{12} & \text{springs not engaged} \end{cases} \tag{22}$$

where $\boldsymbol{\tau}^s \in \mathbb{R}^{12}$, $\mathbf{k}_q \in \mathbb{R}^{12\times12}$, $\boldsymbol{q}^r \in \mathbb{R}^{12}$, and $\boldsymbol{q}^0 \in \mathbb{R}^{12}$ separately denote the spring torques, diagonal (joint-space) spring constant matrix, reference joint angle, and the rest joint angles on the actuated joints, $\mathbf{1}_{12} \in \mathbb{R}^{12}$ and $\mathbf{0}_{12} \in \mathbb{R}^{12}$ are the vectors with the constant value 1 and 0, respectively.

Consequently, the feedforward torque ($\boldsymbol{\tau}^{\text{ff}} \in \mathbb{R}^{12}$) is the sum of GRF torque and spring torque, i.e.,

$$\boldsymbol{\tau}^{\text{ff}} = \boldsymbol{\tau}^{\text{GRF}} + \boldsymbol{\tau}^s. \tag{23}$$

3) Joint-Level Feedback Tracking: A proportional–derivative (PD) controller with low gains is adopted for joint tracking as follows:

$$\boldsymbol{\tau}^{\text{fb}} = \mathbf{K}_p(\boldsymbol{q}^r - \boldsymbol{q}^e) + \mathbf{K}_d(\mathbf{0}_{12} - \dot{\boldsymbol{q}}^e) \tag{24}$$

where $\boldsymbol{\tau}^{\text{fb}} \in \mathbb{R}^{12}$, $\boldsymbol{q}^e \in \mathbb{R}^{12}$, and $\dot{\boldsymbol{q}}^e \in \mathbb{R}^{12}$ separately denote the feedback joint torques, real joint angles, and real joint velocities. $\mathbf{K}_p \in \mathbb{R}^{12\times12}$ and $\mathbf{K}_d \in \mathbb{R}^{12\times12}$ are the diagonal gain matrices.

Finally, the commanded torque ($\boldsymbol{\tau}^r \in \mathbb{R}^{12}$) is generated as follows:

$$\boldsymbol{\tau}^r = \boldsymbol{\tau}^{\text{ff}} + \boldsymbol{\tau}^{\text{fb}}. \tag{25}$$

VII. Simulations Validations

A. Experimental Setup

This part introduces the computing setup for both simulations and hardware experiments.[3]

For jumping motion planning, z_0 is 0.32 m, and the initial joint angles for each leg are [0, 0.857, −1.509] rad. The TO problems in Sections IV-B and V-B are solved using "CasADi" [50] with Python wrapper, taking "IPOPT" [51] as the solver. For jumping motion generation, $N_s = N_f = 100$. The computing time of the offline TO ranges between 150–8000 ms on a laptop with an i7 Intel CPU. For the postimpact LR, $N_l = 10$. It turns out that the computing time is below 30 ms, meeting the real-time requirements. Differing from the TO formulations, the online QP for LMM (in Section V-A) and the MPC (in Section VI-B) are solved in C++, using the open-source "OSQP" [52] solver. The MPC considers seven steps further (i.e., $N_h = 7$) and the average computation cost for each MPC loop is around 2–3 ms. Thus, we set the low-level control frequency to be 333 Hz for both simulations and experiments.

An extended Kalman filter was employed to estimate the system state, including 3-D CoM position and velocity, together with body inclination angles. The foot is considered in contact with the ground when the measured GRF is above a threshold. The contact phase, including the stance phase and landing phase, is then detected when at least two feet come in contact with the ground.

B. SLIP Jumping Performance

1) Jumping Motion Generation: Using the actuated SLIP model in Section IV-A and the KD-type TO in Section IV-B, we generate the optimal jumping trajectory. The resultant trajectories for a diagonal jumping task, e.g., 0.5 m forward and 0.2 m leftward jumping, are plotted in Fig. 8.

Jumping trajectories corresponding to $k_s = 0$ N/m are presented in the first row of Fig. 8, demonstrating that the proposed actuated SLIP model in (2) can be used for a rigid robot without parallel elasticity. In this case, horizontal acceleration inputs (i.e., u_x and u_y) coincide with total accelerations (i.e., $\ddot{x}$ and $\ddot{y}$), see the plots in the second column. When the parallel elasticity is engaged (taking $k_s = 1000$ N/m for an example), the compliant 3-D jumping trajectories are also obtained, as plotted by the second row of Fig. 8. The CoM trajectories in the first column reveal that, in both cases, the final positions are quite close to the targets. Comparison between the vertical CoM trajectories (see the red curves in the first column of Fig. 8) demonstrates that a larger compression during the stance phase with a greater peak height during the flight phase is accomplished when $k_s = 1000$ N/m. That is, parallel compliance is utilized for dynamic jumping.

In addition to the CoM motion, the joint angles and torques of the FL leg are separately plotted in the third and fourth column.[4] As can be seen from the blue curves in the fourth column, the usage of parallel compliance reduces the peak torque on the calf joint.

[3] Videos of all the results are available at https://youtu.be/YMKgi1ro-bM

[4] In this diagonal jumping task, we found the FL leg suffers the biggest GRF and the largest torque among the four support legs.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

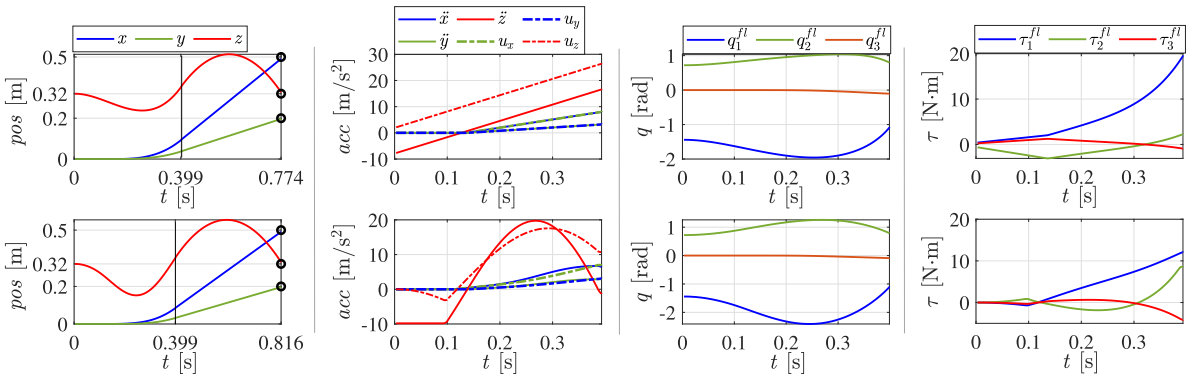

Fig. 8. Jumping data when $k_s = 0$ (the first row) and $k_s = 1000$ (the second row). In the first column, the CoM positions before and after the black lines separately denote the stance and flight trajectories, where the black circles mark the desired landing positions. u_x, u_y, and u_z in the second column separately denote the acceleration input along the x, y, and z axis. $[q_1^{\text{fl}}, q_2^{\text{fl}}, q_3^{\text{fl}}]$ and $[\tau_1^{\text{fl}}, \tau_2^{\text{fl}}, \tau_3^{\text{fl}}]$ separately denote the joint angles and torques of the calf, thigh, and hip joint on the FL leg.

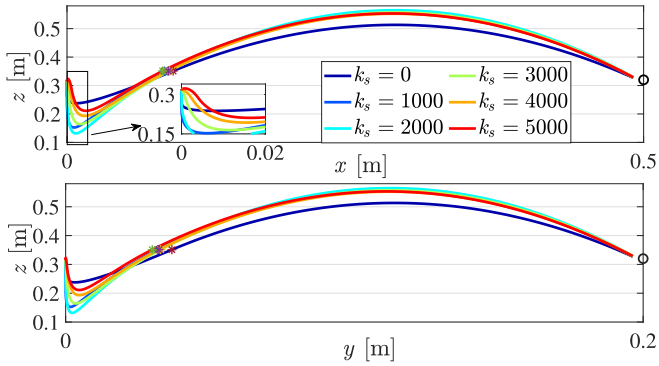

Fig. 9. CoM trajectories corresponding to different spring constants, where the top panel and bottom panel separately plot the sagittal and lateral trajectories. Colorful markers are added to the take-off positions, and black circles mark the desired landing positions.

TABLE I
ENERGETIC PERFORMANCE W.R.T VARIABLE SPRING CONSTANTS

k_s [N/m]	0	1000	2000	3000	4000	5000
$\tau_1^{\max}$ [N·m] (diagonal jumping)	19.5	12.0	8.2	8.9	10.0	11.4
$\boldsymbol{E}$ [J] (diagonal jumping)	40.1	38.9	36.4	35.1	34.5	36.5
$\boldsymbol{E}$ [J] (forward jumping)	36.9	35.8	35.3	33.6	32.6	34.7

2) Energetic Performance With Respect to (w.r.t) Parallel Compliance: For the diagonal jumping task, 3-D CoM trajectories with different spring constants are plotted in Fig. 9. As can be seen from the z curves between the start points and the colorful markers, when using parallel compliance, i.e., $k_s > 0$, the robot can actively compress the springs to store energy and then release it to jump onto the desired location.

Table I evaluates the energetic performance (during the stance phase).[5] As can be seen from the second row, the peak calf torque ($\tau_1^{\max}$) is reduced when the parallel springs are engaged. When $k_s = 2000$ N/m, the peak τ_1^{fl} is reduced by 57.9% (8.2 versus 19.5). Besides, the energy cost during the stance phase drops. Note that the energy cost needed by the forward jumping, i.e., $x^r = 0.5$ m and $y^r = 0$ m, is also computed. In the second column, the total energy cost of the forward jumping is lower than that of the diagonal jumping. Again, smaller $\boldsymbol{E}$s are achieved when $k_s > 0$. The interesting point is that when the parallel springs are too stiff, the energy cost will increase (comparing $\boldsymbol{E}$ at $k_s = 5000$ N/m and $\boldsymbol{E}$ at $k_s = 4000$ N/m). From the partially enlarged drawing on the $x - -z$ curve in Fig. 9, we found that due to the large spring force, the robot increases the height before squatting down. As a result, the robot spends extra energy in accomplishing the jumping task.

[5]The energy cost is computed by $\boldsymbol{E} = \sum_{i=1}^{N} \sum_{j=1}^{12} \tau_j^i \dot{q}_j^i \Delta t$ with τ_j^i and $\dot{q}_j^i$ separately denoting the joint torque and angular velocity of the j-th joint ($j \in \{1, \ldots, 12\}$) at the i-th step, Δt is the time interval for low-level control.

C. Quadrupedal Jumping Simulation

This section validates the jumping performance of the compliant E-Go robot that enhances the Unitree Go1 [20] with parallel springs. In particular, we highlight the effectiveness of the impact-aware LMR strategy. The full-body motions are simulated in *PyBullet* [53], where the spring force on each joint is emulated by a proportional law, i.e., $\tau_j^s = k_j(q_j^r - q_j^e)$ ($j \in \{1, \ldots, 12\}$) with k_j denoting the spring constant of the jth parallel spring. Without extra specification, we set $\delta_x^{\max} = 0.1$ m (for forward jumping), $\delta_y^{\max} = 0.1$ m (for leftward jumping), $z^{\min} = 0.2$ m, and $z^{\max} = 0.42$ m for the preimpact LMM. For postimpact LR, we set $x_l^{\max} = -x_l^{\min} = y_l^{\max} = -y_l^{\min} = 0.05$ m, $z_l^{\min} = 0.15$ m, and $z_l^{\max} = z_0 = 0.32$ m. When generating the quadrupedal leg trajectory, z_{tr} in Fig. 7 is 0.2 m by default. Notably, low-level control gains are unchanged in the following.

1) Tracking Performance: To start, we analyze the jumping performance in executing sagittal jumping tasks.

Jumping with impact-aware LMR: The reference trajectory for 0.7 m forward jumping is generated using the KD-type TO, with $k_s = 1500$ N/m (In PyBullet, the spring constants of the hip, thigh, and calf joint are 6, 15, and 6 to match the hardware design, and the equivalent k_s is then computed by the approach in Appendix B). Results with/without the impact-aware LMR (including preimpact LMM and postimpact LR) are plotted in Figs. 10–12.[6]

[6]When LMR is disengaged, the robot returns to the homing pose by locking joints immediately after detecting landing. Consequently, the actual CoM reference during the landing phase is determined by both the contact center and the homing configuration. However, to emphasize that no landing motion is planned online in this case [noted by "(no LMR)" in Figs. 10–12], we here still use the offline-generated jumping trajectory as the reference for the "(no LMR)" jumping. In contrast, when the LMR is engaged, the reference landing motion will be modulated in real-time and then be updated in the plots.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

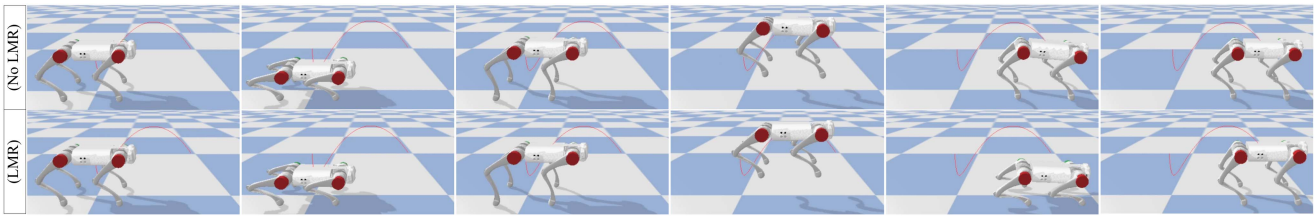

Fig. 10. Quadrupedal robot jumps on a flat ground, where the red curve plots the reference trajectory generated by the offline TO. The first row shows the jumping motion without LMR (i.e., without LMM+LR), whereas the second row shows the jumping motion using LMR (i.e., with LMM+LR).

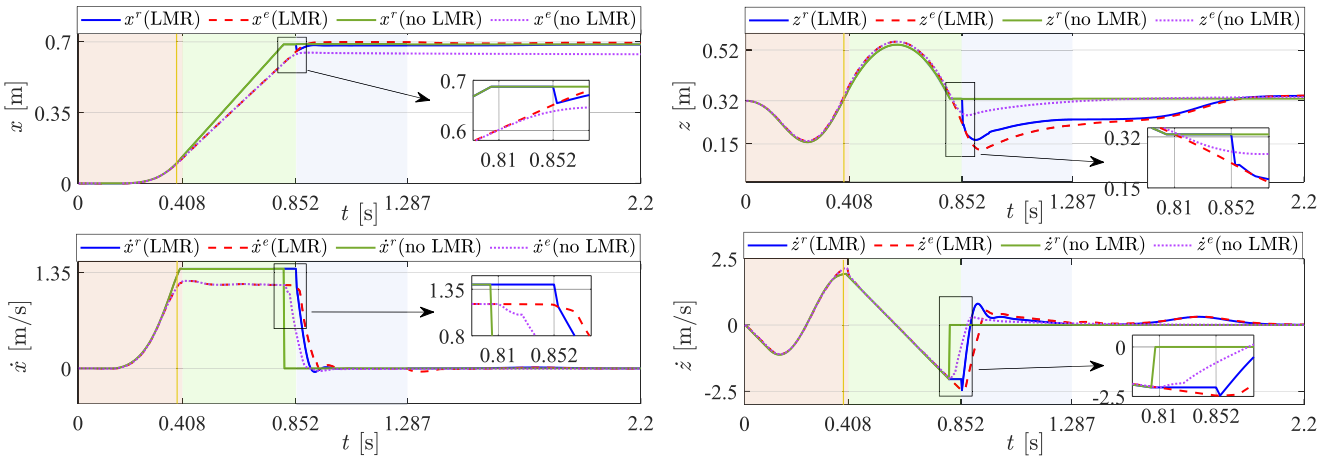

Fig. 11. Quadrupedal robot jumps forward on the flat ground. "(LMR)" and "(no LMR)" separately denote the results with and without the impact-aware LMR. The shallow red, green, and blue stripe zones separately cover the real stance, flight, and recovery motions when using LMR. The robot motions around the real landing moments are detailed in the partially enlarged drawings. The yellow line marks the reference take-off time.

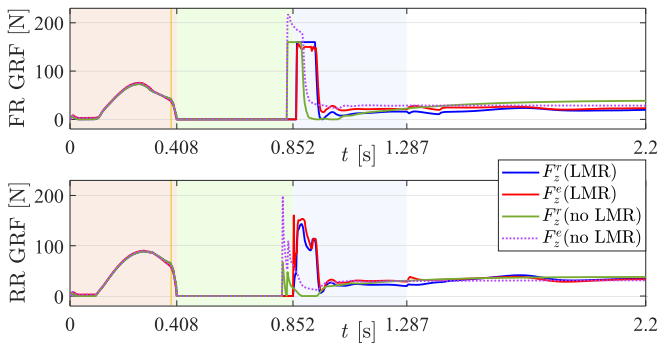

Fig. 12. Verical GRF profiles for the forward jumping on the flat ground. "F_z^r" and "F_z^e" separately denote the reference and measured vertical GRF. The yellow line marks the reference take-off time.

The reference trajectories (the green curves) in Fig. 11 reveal that the robot would takeoff at 0.399 s (marked by the yellow line) and land at 0.804 s. Due to the modeling errors, the robot tracked the reference during the stance phase with errors. As a result, the robot took off later, considering that the measured vertical GRFs ("F_z^e" curves) in Fig. 12 become zeros at 0.408 s. When the LMR strategy was not integrated [see "(no LMR)" curves], the tracking errors at the take-off moment led to an undesired landing status. Specifically, the smaller forward position [see "x^e(no LMR)" versus "x^r(no LMR)" at 0.408 s] with a smaller forward velocity [see "$\dot{x}^e$(no LMR)" versus "$\dot{x}^r$(no LMR)" at 0.408 s] led to a shorter landing position. It turns out that the robot landed at 0.647 m [check the "x^e(no LMR)" plot after 0.9 s in Fig. 11].

In contrast, when the preimpact LMM was incorporated, the robot moved the leg forward (optimized δ_x is 0.035 m) with retraction (optimized δ_z is 0.2 m) in the air, as demonstrated in the fourth and fifth snapshots in the second row of Fig. 10. As a consequence, the robot landed later at 0.852 s, as evidenced by the "F_z^e(LMR)" curve in the bottom panel of Fig. 12. And the robot stopped at 0.695 m, which is close to the desired location. After touchdown, the robot followed the updated reference trajectories that are generated by the postimpact LR, i.e., the online TO in Section V-B, see the blue-solid and red-dashed curves in Fig. 11 after 0.852 s. Note that the final CoM height generated by the online LR is 0.247 m, see the "z^r(LMR)" at 1.287 s. Thanks to the polynomial interpolation, the robot eventually returned to the homing height. Since the estimated support height when landing is 0.02 m, the final body height is 0.34 m.

Vertical GRFs on the FR and RR legs are plotted in Fig. 12. When LMR was not engaged, the robot touched down with a sudden brake, resulting in larger peak GRFs, as plotted by "F_z^e(no LMR)" curves. In the "(no LMR)" case, since the robot tried to decrease the velocity to zero without any reconciling motion, the front leg encountered a larger GRF than the rear leg. In contrast, with LMR, a compliant landing with smaller peak GRFs is realized, and the body weight is finally uniformly distributed into the front and rear legs.

Furthermore, Fig. 13 highlights the effect of the postimpact LR by comparing the pitch angle and torque profiles (in the

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

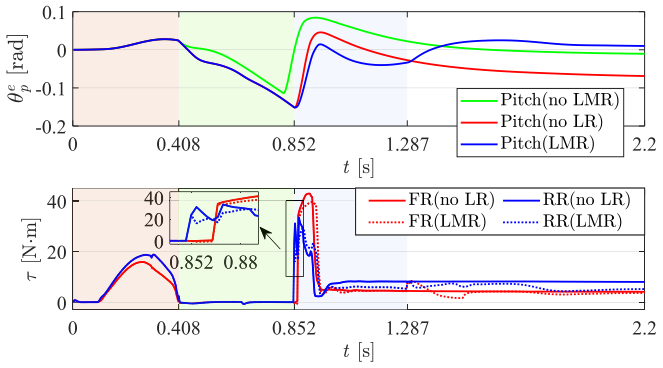

Fig. 13. Body pitch (top) and right leg knee torque (bottom) for the forward jumping on the flat ground. Particularly, the jumping using LMM but without LR scheme is also detailed, denoted by "(no LR)."

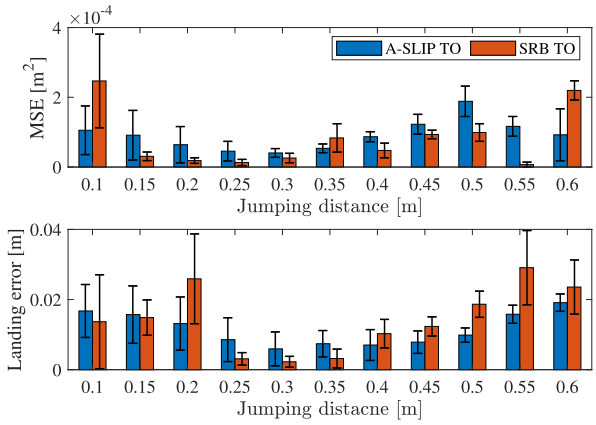

Fig. 14. Comparison study of tracking errors. From top to bottom, the mse of the forward movement and the absolute error of the forward landing position are separately displayed.

knee joint) between "(LMR)" and "(no LR)." When LR was not integrated, the robot locked its joint to maintain *M-land* configuration. In this case, the robot leaned backward after landing, considering that the legs were driven forward in the air to reach the desired landing position. Also, as mentioned in Section VI-A, the body pitch is not forced to return back to zero when LR is not used. As a result, there is a negative body pitch after landing, see "Pitch(no LR)" curve after 1.287 s in Fig. 13. In addition, since no reconciling motion was imposed in the "(no LR)" case, the knee suffered larger torques right after touchdown, as evidenced by the partially enlarged drawing. Then, the rear legs supported most of the body weight by providing a larger torque, see the "RR(no LR)" curve after 2 s. In contrast, in the "(LMR)" case, the weight was uniformly distributed into the rear and front legs.

Aside from forward jumping, the robot could accomplish other versatile jumping tasks, e.g., backward 50 cm jumping and 3-D jumping onto the 20 cm high table. All the tests reveal that the LMR scheme helps to realize a stable landing with quick recovery. To save space, we put the comparison results into the attached video of the Supplementary Material.

Comparison studies with an SRB TO: We here compare the proposed KD-type TO with a standard KD TO [15], where the SRB model is used. In the baseline (noted as "SRB TO"), in addition to body movement and joint movement, the leg movement and GRF are also optimized.[7] In each simulation, we added random noise in the sensory feedback (including CoM position, CoM velocity, body angle, and angular velocity) and calculated the tracking errors in different channels, including the mean square error (mse) of forward movement before the landing phase and the absolute tracking error of forward landing position. Statistical results of 20 trials under each jumping distance are reported in Fig. 14.

Fig. 14 reveals that "SRB TO" results in smaller tracking errors than our KD-type TO (noted as "A-SLIP TO") in some cases but larger errors in other cases. That is, our TO obtained a comparable result with the standard KD optimizer. Note that the online LMM is not integrated here for a fair comparison. The result in Fig. 11 has demonstrated that LMM would achieve a more accurate tracking of the landing position.

[7]Without lifting off the front legs during the stance phase, the "SRB TO" hardly generates motions for longer jumps than 0.6 m.

2) Robustness Against Uncertain Landing Surfaces: This section validates the landing robustness against external disturbance, e.g., surface uncertainties. In this test, the reference trajectories for 1 m forward jumping are generated by the offline KD-type TO. Then, the robot jumps toward an unknown ramp (inclination angle is 18°), which is placed at 0.75 m in front. Comparative results are presented in Figs. 15–17.

The reference jumping trajectories expect the robot to land at 0.826 s. When the LMM was disengaged, the robot landed earlier at 0.807 s (see the pink dot curves in the partially enlarged drawings in Fig. 15), with front legs touching the ground [see "F_z^e(no LMR)" in the top panel of Fig. 16]. After touchdown, the robot tracked the homing configuration without exploring a reconciling recovery motion. As a result, the robot suffered from a larger peak force in the front and rear legs, which can be found when comparing "F_z^e(no LMR)" with "F_z^e(LMR)" in Fig. 16. Besides, due to the surface inclination, the CoM would lean behind the center of the support feet during the landing phase. Consequently, after bouncing, the robot tipped over. The falling robot is shown in the seventh picture in the first row of Fig. 17.

On the contrary, when LMM was engaged, the robot moved the leg forward (δ_x is 0.1 m) in the air with a retraction motion (δ_z is 0.2 m), as can be seen from the fourth and fifth pictures at the bottom of Fig. 17. Consequently, although there is a convex surface, the later landing still happened (landed at 0.831 s in this scenario). When the LR was not activated, the robot landed stably but ended with a lower height and large body inclination. In contrast, with LR, the robot generated reconciling recovery motions after landing (see blue curves covered by shallow-blue stripe in Fig. 15), resulting in lower peak GRFs [plotted by "F_z^e(LMR)" in Fig. 16]. When detecting landing, the robot squatted down rapidly. After reaching the lowest height, the rear leg pushed the robot forward and later drove the robot upward. As a result, the rear leg suffered a larger GRF than the front leg, as demonstrated by checking "F_z^e(LMR)" for the RR and the FR legs. Note that due to the landing impact and the reducing body height (needed by the recovery motion) right after touchdown, the legs lifted off for a short period, see "F_z^e(LMR)" from 1–1.1 s. Nevertheless, the robot with retracted legs stabilized itself using the hybrid torque control scheme. Finally, the robot

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

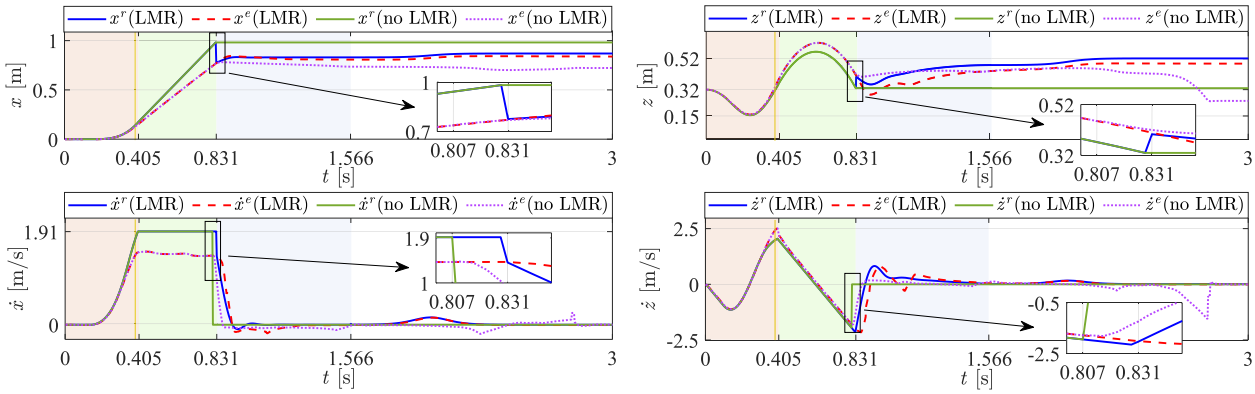

Fig. 15. Quadruped jumps onto an unknown slope. "(LMR)" and "(no LMR)" separately denote the results with and without the impact-aware LMR. The robot motions around the real landing moments are detailed in the partially enlarged drawings. The yellow line marks the reference takeoff time.

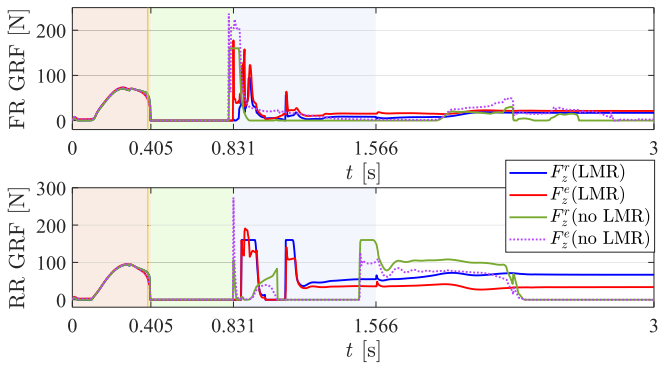

Fig. 16. Vertical GRFs for quadrupedal jumping onto an unknown slope. The yellow line marks the reference takeoff time.

regulated its CoM to lie above the support center, resulting in a stable recovery (see the second row in Fig. 17).

Aside from this, the robot also managed to jump on a rough surface with a random height profile. Taking the 0.7 m forward jumping as an example, comparison studies (in the attached video in Supplementary Material) demonstrate that when jumping across an uneven surface with random height (Gaussian height map with ± 1.5 cm in height variation), the robot landed at 0.67 m by using LMR, longer than 0.61 m when not using LMR.

3) Jumping With Unknown Obstacles: Adaptable jumping in scenarios with unknown convex obstacles is also validated in simulations. To avoid collisions with these convex objects, the robot should jump as high as possible while retracting the leg. To increase the peak height in the air, we can use a smaller w_f or set a larger z_t in (4d). In addition, a smaller z_{tr} in Fig. 6(a) and a smaller height boundary $z^{\min}$ in (16) could be chosen to enable a larger retraction. In this section, 1 m forward jumps under two scenarios are tested, including jumping onto a box with an unknown height and jumping above an obstacle. The resultant jumping motions are visualized in Fig. 18.

In the first scenario (the top of Fig. 18), the jumping trajectory was generated through setting a smaller w_f in (4d). Meanwhile, we dropped z_{tr} from 0.2 m to 0.1 m. As a result, the peak height of the reference trajectory increased from 0.56 m to 0.59 m. By retracting the leg, the robot could land on a 0.3 m high table placed at 0.5 m in front.

In the second case, we validate adaptable jumping above a convex obstacle. To encourage a higher jumping, we increased z_t [in (4d)] from 0.39 m to 0.5 m. Besides, the transition height z_{tr} was reduced to 0.05 m (within the kinematic limit), and $z^{\min}$ was dropped to 0.15 m. As a result, the robot successfully jumped above a 23 cm high board, see the bottom of Fig. 18.

In contrast, without LMR, the robot failed in the above two tasks. Without LR, the robot landed with an undesired ending pose. Please check the video for detailed comparisons in Supplementary Material.

VIII. Hardware Experiments

This section presents the hardware experiments on the E-Go robot, which enhances the rigid robot with parallel compliance. To start, we briefly introduce the design of elasticity add-ons. Then, we extensively validate the proposed approach with a rigid robot. Afterward, we highlight the enhanced jumping performance by exploiting parallel compliance. Without different specifications, the open parameters for offline TO, online LMM, and LR used in hardware tests are the same as those used in simulations, except that $z^{\min}$ and $z_l^{\min}$ are increased to 0.25 m and 0.2 m separately to avoid collisions with the ground. In addition, the following jumping tasks share the control gains. Inspired by [9], we reduced the PD gains for (24) after touchdown to enable a compliant landing.

A. E-Go Design

Delft E-Go is an articulated soft quadruped robot, which is made by enhancing the commercially available robot Go1 from Unitree [20] with a set of mechanical add-ons. In E-Go, a dedicated parallel spring strengthens each actuated joint. As illustrated in Fig. 19(a), the thigh joint is strengthened by cable-driven PEA (visualized in the left column of Fig. 20) and the calf joint is strengthened by a mono-articulated PEA (see the right column of Fig. 20). Particularly, the locking mechanism (see Fig. 20) in Delft E-Go encourages easy engagement/disengagement of the parallel springs, enabling a switch between a rigid and soft configuration. Also, users can replace the spring with different constants. These properties support extensive hardware studies with this platform.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

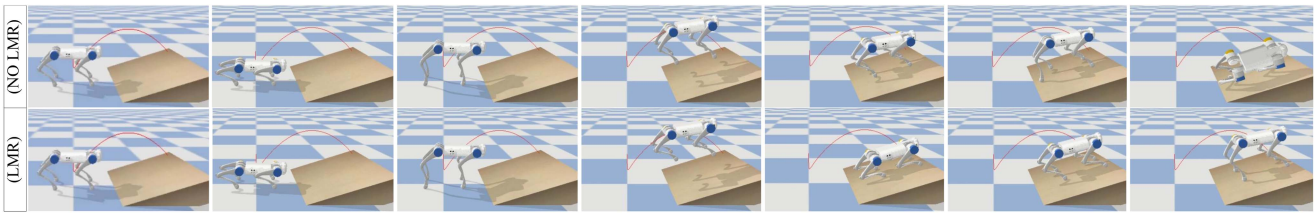

Fig. 17. Quadruped jumps onto an unknown slope. The first row shows the jumping motion without LMR (i.e., without LMM+LR), whereas the second row shows the jumping motion using LMR (i.e., with LMM+LR).

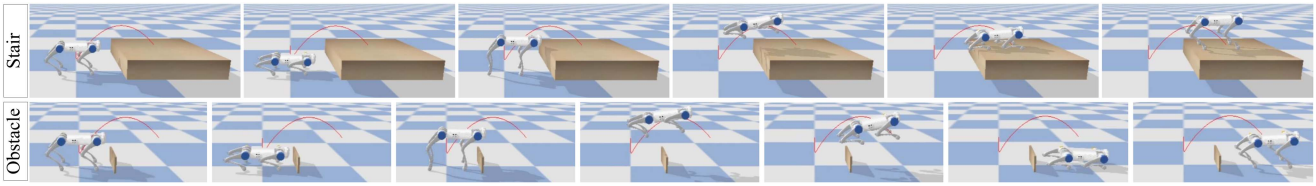

Fig. 18. Jumping with unknown convex obstacles. (Top) Jumping onto a table. (Bottom) Jumping above a board. At the top, one 1 m×1 m×0.3 m table was placed at 0.5 m in front. At the bottom, the 2 cm thin board was placed 0.15 m ahead.

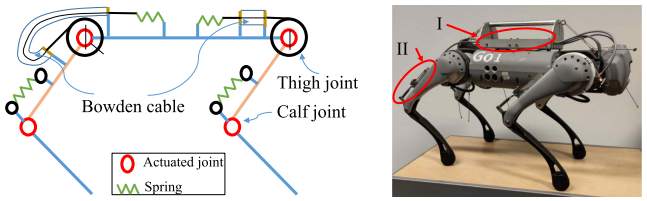

Fig. 19. (a) PEA design of E-Go. (b) Hardware platform.

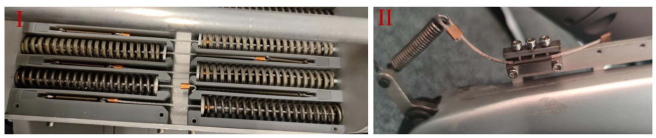

Fig. 20. Visualizations of spring configuration for thigh joint (left column) and the PEA for the calf joint (right column).

B. Jumping With Impact-Aware Landing

In this section, we focus on the jumping control of a rigid quadruped when the parallel springs are disengaged.

1) Controlled Jumping on Flat Ground: First, jumping performance is compared when incorporating the impact-aware LMR scheme or not. Fig. 21 captures the forward jumping motions (jumping for 0.5 m on the flat ground) under different conditions[8] and Figs. 22 and 23 plot the resultant trajectories.

The yellow line in Fig. 22 marks the reference take-off time. As can be seen from Fig. 22, large tracking errors occurred in the vertical (z) movement, resulting in a later takeoff. When not regulating the landing motion ["(no LMR)" case], the robot landed earlier due to the poor shooting states, e.g., a lower vertical height at 0.477 s that is plotted at the bottom of Fig. 22. In this case, the real flight duration is 0.306 s, and the estimated landing position is 0.42 m (see the partial drawing in Fig. 22 and the fifth picture in the first row of Fig. 21). In contrast, when modulating the leg motion in the air, i.e., using the preimpact LMM, E-Go moved the leg ahead with a retraction motion (comparing the fourth with the fifth picture in the second and third rows of Fig. 21). As a result, the flight period increased, resulting in a longer flight distance. Specifically, in the "(LMR)" case, the flight phase increased to 0.33 s (see the green zone in Fig. 22), and the robot landed at 0.495 m (see the final "x^e(LMR)" in Fig. 22).[9]

In the first row of Fig. 21, the postimpact LR was not integrated either. That is, the robot aimed for the *D-land* configuration right after touchdown. As a result, the robot hit the ground heavily, with a large body rotation (see fifth picture in the first row of Fig. 21 and the pink dotted curve after 0.8 s in Fig. 23). Due to the large impact, the robot bounced upward. Meanwhile, the robot continued moving forward, resulting in a second "jumping," which can be seen when comparing the landing positions in the fifth and sixth pictures in the first row of Fig. 21. In contrast, when only integrated LMM, the robot landed at the desired position in the *M-land* configuration [see the fifth picture in the "(No LR)" case]. Under the compliant torque control, the robot then reduced the height to dissipate the kinetic energy [see the sixth picture in the "(No LR)" case]. Similar to the forward jumping test in Section VII-C1, there is also a negative body pitch. When using LR "(LMR)", the robot first squatted down after detecting landing, followed by a reconciling recovery motion, as plotted by the x and z motions after 0.807 s in Fig. 22. Finally, the robot returned to the homing state, i.e., *D-land* configuration, as evidenced by the sixth picture in the third row of Fig. 21. Aside from this, the robot almost returned to zero pitch when LR is used, as plotted in Fig. 23.

More tests revealed that, without LMR, the landing error would increase as the desired jumping distance rises. Instead, with LMR, a longer jumping distance could be reached. The

[8]In real tests, the sensory noise could result in large estimation errors in the global landing position. To provide an alternative solution, we put white markers on the ground. The distance between the sparse markers is 20 cm while the distance between dense markers is 10 cm.

[9]Fig. 21 tells that the robot in the "(no LR)" and "(LMR)" cases both landed on the desired positions, demonstrating our LMM scheme could achieve reliable tracking performance in repetitive tests.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

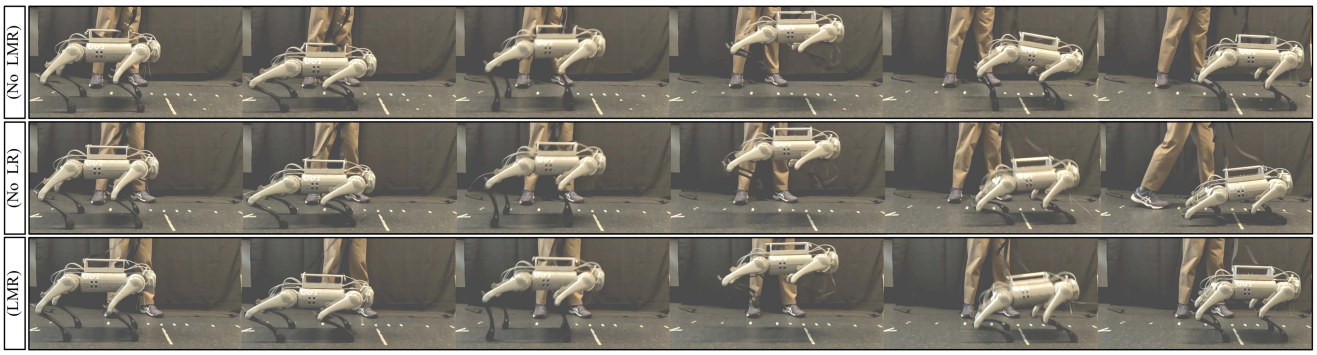

Fig. 21. Jumping motions on flat ground, using the rigid quadruped without springs engaged. The first, second and third rows separately demonstrate the jumping motion without LMR (i.e., without "LMM+LR"), without LR (i.e., only use "LMM") and with LMR (i.e., with "LMM+LR").

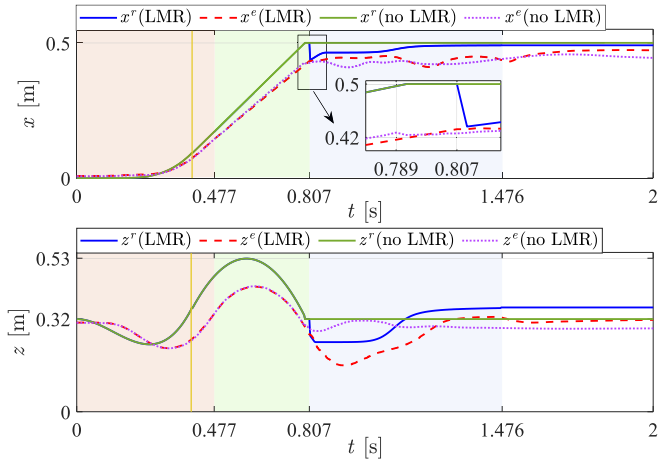

Fig. 22. CoM trajectories for 0.5 m forward jumping. The yellow line marks the reference take-off time. We omit the trajectories in the "(no LR)" case to keep this figure clean. At the beginning of LR, the "z^r(LMR)" was clamped by the lower boundary.

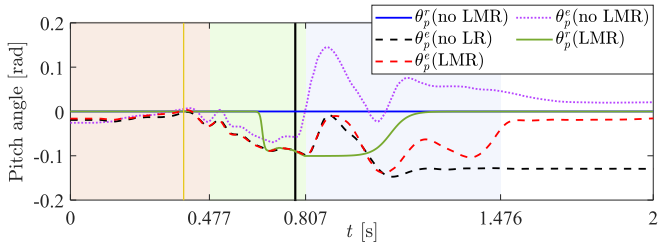

Fig. 23. Pitch angles for 0.5 m forward jumping. The yellow line marks the reference take-off time, and the bold black line marks the real landing time without LMR.

0.6 m forward jumping with/without LMR can be found in the attached video in Supplementary Material.

We noticed that the front legs bounced after touchdown when using the LR scheme, which can be explained as follows. First, the reduced PD gains for (24) resulted in softer legs after landing. Considering that the reference CoM (from *M-land* to *D-land*) fell behind the support center, the robot was prone to lean backward during the recovery process. Then, as the body height increased, the tilting momentum rose, and the front legs lost contact with the ground. Nevertheless, the low-level controller managed to stabilize the robot.

2) 3-D Jumping: With the LMR scheme, the rigid robot can achieve versatile jumping tasks, such as 50 cm backward, leftward, and rightward jumping. Here, we present leftward jumping in Fig. 24(a). As plotted by Fig. 24(a), there is a large rotational movement in the air when jumping left. Nevertheless, with LMR, the robot recovered to the *D-land* configuration with small inclinations.

By changing the desired landing height, i.e., h_d in (7), jumping onto a high box is also realized. In Fig. 24(b), we demonstrate 60 cm forward jumping with a stable landing onto a 7 cm high pad. Furthermore, in the limit test, the robot could jump onto the 10 cm pad. However, in this case, the robot landed with a large touch-down angle, where the contact of the rear legs was not measured. As a result, the landing event was not detected, and the postimpact LR was not activated. The detailed motion can be found in the attached video in the Supplementary Material.

Diagonal jumps are also achieved using the proposed approach, one of which is demonstrated in Fig. 1(a).

C. Robust Jumping

The above results have demonstrated that our approach is robust against system uncertainties such as modeling errors caused by model simplifications. In this section, we further validate the robustness against external disturbances.

1) Robust Jumping Against CoM and Angular Offsets: In the offline TO formulation, we assume the robot starts in the homing configuration where the CoM lies above the support center with zero inclinations angles (see Fig. 4). However, the above assumption is usually violated in real scenarios. To demonstrate robustness against the initial state deviation, we make the robot jump off of the unknown object, i.e., starting with CoM and angular offsets. It turns out that, using the proposed approach, the robot could still accomplish jumping tasks with stable landing and quick recovery. In Fig. 25(a), the robot jumped off of a 4 cm-high pad and landed 40 cm ahead on the slippery paper board. In addition, the robot could jump off of a 7 cm-high pad and land 45 cm behind, see the attached video in Supplementary Material. Note that a similar test has been found in [13], where, however, a RL strategy was additionally required to train the control policy by mimicking the reference. In that work, backward jumping was not shown either.

2) Robust Jumping Against External Forces: Stable jumping under external pushes/pulls is a challenging task where the robot needs to overcome huge deviations in the linear and angular velocity. In this section, external forces are imposed after the

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

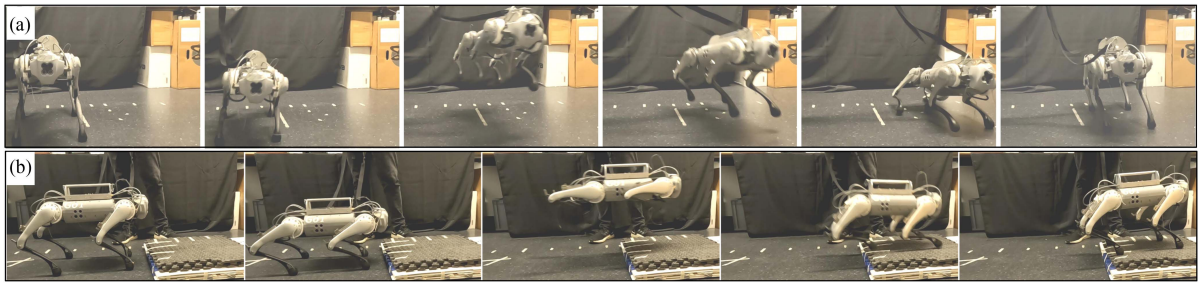

Fig. 24. E-Go (without springs engaged) jumps with impact-aware LMR for different 3-D tasks. The pictures separately illustrate (a) 0.5 m leftward jumping and (b) 0.6 m forward jumping onto a 7 cm high pad.

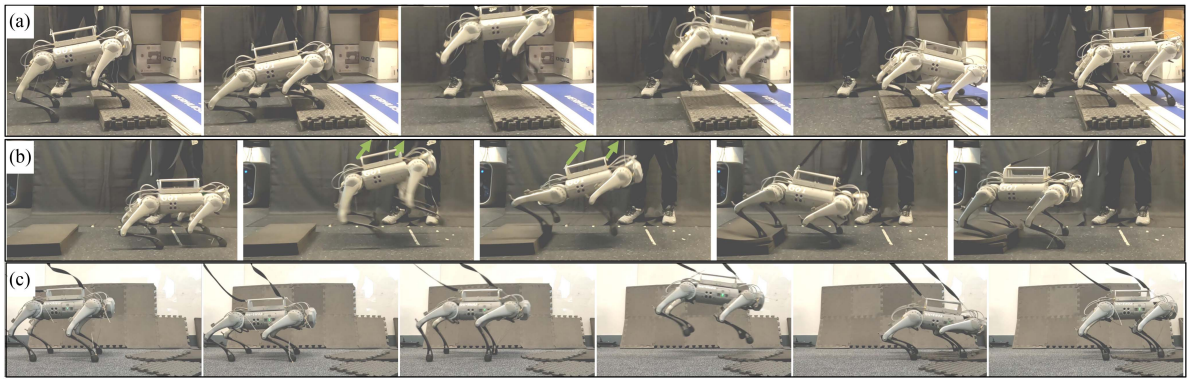

Fig. 25. E-Go (without springs engaged) jumps against external disturbances with the LMR scheme. (a) Robot jumps off of the 4 cm pad in the forward direction. (b) Robot jumps against external forces. (c) Robot jumps onto an uneven surface. In (b), the green arrows mark the force directions.

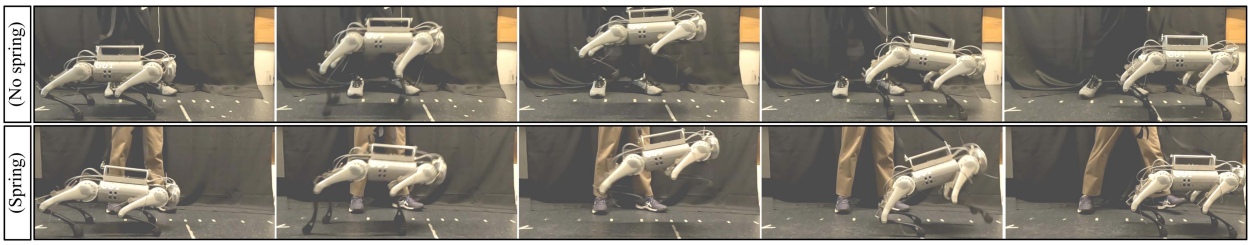

Fig. 26. E-Go jumps for 0.6 m. (Top) Rigid robot jumps without springs engaged. (Bottom) Compliant robot jumps with springs engaged. Note that the LMR scheme is not activated in both cases.

robot takes off, see the green arrows in Fig. 25(b). It turns out that the E-Go robot could still land stably with quick recovery since the postimpact LR modulates the landing motion in real-time based on real landing status. And, the low-level torque control strategy realized a compliant tracking. As can be seen in the fourth and fifth pictures of Fig. 25(b), the robot recovered from a large inclination angle when landing on an uneven surface. In addition, leftward jumping under external pushes was achieved, as shown in the attached video in Supplementary Material. Notably, a similar test was found in [54] where external pushes were imposed when the biped jumps in place. However, in [54], the robot landed on a flat surface. In addition, no side jumping against external forces was reported there.

3) Robust Landing on Uneven Surfaces: Differing from the landing controller in [17], our impact-aware LMR scheme did not assume that the robot lands on flat ground. As a result, our robot can jump onto uneven surfaces with variant height. One of the tests is already shown in Fig. 25(b), where the robot landed backward on the unexpected 3 cm soft pad. In Fig. 25(c), the quadruped landed stably on the uneven surface with rigid pads placed at random heights.

D. Jumping With Parallel Springs

1) Explosive Jumping: When engaging the parallel springs, explosive jumping achieving a greater forward distance and landing height could be realized. In the current test, we found that, without LMR, at least 0.6 m forward jumping could be realized, resulting in a 20% increase in jumping distance, considering that the rigid robot can only jump to 0.5 m without parallel springs. Fig. 26 captures the 0.6 m forward jumping motions, where the first and second rows separately demonstrate the motions without/with spring engaged. As shown in the first row, the rigid robot without springs landed at about 45 cm (see the fourth picture) and then bounced forward. In contrast, the compliant robot landed exactly at 60 cm. The second picture at the bottom of Fig. (26) reveals the rear legs slipped during the

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

TABLE II
ENERGY COST W.R.T SPRING CONSTANT VARIATION

k_s [N/m]	0	1500	2300	3200
$\tau_1^{\max}$ [N·m] (diagonal jumping)	33.0	27.0	34.9	32.4
$\boldsymbol{E}$ [J] (diagonal jumping)	66.4	60.9	75.3	64.5
$\boldsymbol{E}$ [J] (forward jumping)	62.5	57.4	71.5	63.1

stance phase. Nevertheless, our controller is robust enough to achieve a stable landing.

Furthermore, by exploiting parallel elasticity, the robot successfully jumped onto a 15 cm stair with a recovery motion [see Fig. 1(c)], resulting in a 50% increase in landing height considering that the rigid robot could only jump onto a 10 cm stair. Comparison studies can be found in the video in Supplementary Material.

2) Energetic Benefits: By changing the springs on the calf joints, the energy cost of the jumping motion w.r.t parallel compliance is reported. Table II evaluates the energy performance for two tasks, i.e., 50 cm forward jumping and 50 cm forward plus 20 cm leftward jumping. Taking diagonal jumping for an example, by choosing the proper parallel springs, e.g., $k_s = 1500$ N/m, the peak calf torque is reduced by 18.2% whereas total energy cost drops by 8.3%.

3) Robust Jumping: Like the rigid case, the compliant quadruped with parallel springs is also robust to external disturbances, including ground unevenness, external force, and CoM and angular offsets. Please check the video in Supplementary Material.

Notably, Fig. 1(b) demonstrates that the robot could jump from a 5 cm block, which is higher than that of the rigid case, i.e., 4 cm in Fig. 25(a). This is because the parallel springs enhance the knee joints on the rear legs by providing additional torques. In that case, all the support legs can keep in touch with the surface during the stance phase.

IX. CONCLUSION

In this work, we address 3-D quadrupedal jumping exploiting parallel compliance. To start, we introduce a novel actuated SLIP model. Differing from previous SLIP and its variants, the proposed model is able to explicitly characterize the passive elasticity introduced by the parallel springs, enabling us to model both rigid and soft articulated quadrupedal robots. Subsequently, based on the reduced-order model, we formulate an offline trajectory optimizer to generate 3-D jumping motions. In addition, we propose an online impact-aware LMR strategy, including preimpact leg motion regulation and postimpact LR, resulting in a stable landing with compliant recovery.

Using the proposed approach, we realize robust 3-D jumping for both rigid and compliant quadrupedal robots, with the capability of rejecting dynamic disturbances such as modeling errors, ground unevenness, and push forces. Particularly, the hardware experiments demonstrate explosive and energy-efficient jumping by exploiting parallel elasticity.

The below discussions draw connections with related works.

A. Landing With Variable Stiffness

It has been demonstrated the postimpact LR scheme helps to realize quick recovery with compliance. However, when jumping at a long distance or landing on uneven terrain, the legs could hop or lose contact with the ground during the recovery process, as mentioned in Section VII-C2 and Section VIII-B1. Although such landing behavior is evidenced in many studies, such as [9] and [24], this kind of leg movement could be risky.

To mitigate the hopping motion, a more thorough analysis of the system's stiffness and damping property is needed. Ideally, critically damping behavior during the landing phase is desired. To this end, we can modulate the spring constant of the SLIP model in the landing phase. Besides, we can modulate the stiffness and damping in the low-level control scheme [55], [56]. Our preliminary study reveals that the variable stiffness and damping control approach in [56] helps to mitigate the leg's hopping after landing.[10] In the future, it would be interesting to investigate the problem fully.

B. Reactive Steps After Landing

In legged robotics, capturability analysis [57], [58], [59] provides a powerful tool to understand how many steps are needed before coming to a stop. In the current work, we assume that the quadruped could stop without making an extra step. That is, the landing behavior obeys zero-step capturability. However, with large landing velocities, the robot may need to take one or several reactive steps before stopping.

Following this idea, the robot could make use of "consecutive" steps to balance aggressive landing velocities. For example, when jumping above a convex obstacle (see the bottom of Fig. 18), one consecutive jump of a short distance could be performed after landing. Fortunately, this kind of motion can be generated using our motion planner by changing the initial state in (5). One initial trial is attached in the video in Supplementary Material. In the future, with a motion library, the robot could automatically pick consecutive motions after a challenging jumping, obeying capturability dynamics.

C. Momentum-Aware Optimization and Control

In the current motion planner, the actuated SLIP ignores the body rotation and leg mass. When modulating the leg motion in the air, we assume no change in the inertial or angular momentum. So does the low-level controller. However, the large leg movement could result in variation in momentum inertia due to the mass distribution. As a result, the robot could suffer from a large body rotation in the air. For example, Fig. 13 reveals that leg movement in the air [in the "(no LR)" and "(LMR)" cases] indeed changes the pitch angle.

Existing works, such as [24], [35], [43], and [44] demonstrate angular momentum adaptation or "inertial shaping" plays a crucial role in achieving dynamic locomotion or jumping tasks. Particularly, the work in [16] argues that a rotating body helps to land stably despite a large landing velocity. To enhance our work, it would be interesting to integrate angular momentum optimization in the motion planning stage. Besides, a full-body controller accounting for leg dynamics can be integrated for better tracking.

In addition, our work can be improved by optimizing PEA parameters such as spring stiffness. We leave it as future work.

[10] Check the attached video for more details.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

APPENDIX A
LAGRANGIAN DYNAMICS FOR ACTUATED SLIP

Considering a legged robot with massless legs, the jumping motion is parameterized by the generalized coordinates $\boldsymbol{c}$. Then, we have the following equation of motion (EoM):

1) Stance Dynamics: During the stance phase, the contact wrench ($\boldsymbol{\mathcal{F}} \in \mathbb{R}^3$) resulted from torque inputs would drive the robot. Considering the nonlinear compliance dynamics from passive elasticity, the EoM is derived obeying Lagrangian mechanics, which is as follows:

$$\mathbf{M}(\boldsymbol{c})\ddot{\boldsymbol{c}} + \boldsymbol{G}(\boldsymbol{c}) + \mathbf{K}(\boldsymbol{l}(\boldsymbol{c}))|\boldsymbol{l}_0(\boldsymbol{c}) - \boldsymbol{l}|\hat{\boldsymbol{l}} = \boldsymbol{\mathcal{F}} \tag{26}$$

where $\mathbf{K}(\boldsymbol{l}(\boldsymbol{c})) \in \mathbb{R}^{3\times3}$ is the time-varying spring stiffness matrix and $\boldsymbol{l}_0(\boldsymbol{c})$ is the time-varying rest length.

Here, the time-varying $\mathbf{K}(\boldsymbol{l}(\boldsymbol{c}))$ and $\boldsymbol{l}_0(\boldsymbol{c})$ are introduced to capture the variation of the rest length [30], [31], [32], spring constant [33], or spring force rules [34]. By linearizing the $\mathbf{K}(\boldsymbol{l}(\boldsymbol{c}))$ and $\boldsymbol{l}_0(\boldsymbol{c})$ around the equilibrium point, we can decompose the nonlinear term on spring force ($\mathbf{K}(\boldsymbol{l}(\boldsymbol{c}))|\boldsymbol{l}_0(\boldsymbol{c}) - \boldsymbol{l}|\hat{\boldsymbol{l}}$) into a linear component ($\mathbf{K}_s|\boldsymbol{l}_0 - \boldsymbol{l}|\hat{\boldsymbol{l}}$) and a general nonlinear component ($\boldsymbol{f}(\mathbf{K}_s, \boldsymbol{l})$), resulting in the following:

$$\mathbf{M}(\boldsymbol{c})\ddot{\boldsymbol{c}} + \boldsymbol{G}(\boldsymbol{c}) + \mathbf{K}_s|\boldsymbol{l}_0 - \boldsymbol{l}|\hat{\boldsymbol{l}} + \boldsymbol{f}(\mathbf{K}_s, \boldsymbol{l}) = \boldsymbol{\mathcal{F}} \tag{27}$$

where $\mathbf{K}_s$ is the constant stiffness matrix and $\boldsymbol{l}_0$ are the constant rest leg length. By introducing the equivalent acceleration $\boldsymbol{u}$ such that $\mathbf{M}(\boldsymbol{c})\boldsymbol{u} = -\boldsymbol{f}(\mathbf{K}_s, \boldsymbol{l}) + \boldsymbol{\mathcal{F}}$, (27) can then be rearranged as follows:

$$\mathbf{M}(\boldsymbol{c})\ddot{\boldsymbol{c}} = -\boldsymbol{G}(\boldsymbol{c}) - \mathbf{K}_s|\boldsymbol{l}_0 - \boldsymbol{l}|\hat{\boldsymbol{l}} + \mathbf{M}(\boldsymbol{c})\boldsymbol{u}. \tag{28}$$

Considering the lumped mass at the body center, $\mathbf{M}(\boldsymbol{c})$ and $\boldsymbol{G}(\boldsymbol{c})$ are independent from $\boldsymbol{c}$. That is, $\mathbf{M}(\boldsymbol{c}) = \mathbf{M} = m\mathbf{I}_3$ and $\boldsymbol{G}(\boldsymbol{c}) = -m\boldsymbol{g}$. Then, we obtain the same stance dynamics with the first row in (2).[11] Note that spring constant $\mathbf{K}_s = -k_s\mathbf{I}_3$ and rest length $\boldsymbol{l}_0$ can be identified with the robot in the equilibrium state, i.e., the homing pose in this work.

2) Flight Dynamics: During the flight phase, the system follows a ballistic trajectory, whose EoM is determined as follows:

$$\mathbf{M}(\boldsymbol{c})\ddot{\boldsymbol{c}} + \boldsymbol{G}(\boldsymbol{c}) = \mathbf{0}. \tag{29}$$

Considering $\mathbf{M}(\boldsymbol{c}) = m\mathbf{I}_3$ and $\boldsymbol{G}(\boldsymbol{c}) = -m\boldsymbol{g}$, we then obtain the flight dynamics expressed in the second row in (2).

APPENDIX B
MATCHING SPRING CONSTANT

Assuming a small variation in the actuated joint angles of each leg, the potential energy ($V_{\text{quad}}^{\text{spring}}$) contributed by spring deformation is calculated as follows:

$$V_{\text{quad}}^{\text{spring}} = \sum_{j=1}^{j=4}\left(\frac{1}{2}(\delta\boldsymbol{q}^j)^{\mathrm{T}}\mathbf{k}^j(\delta\boldsymbol{q}^j)\right) \tag{30}$$

where $\delta\boldsymbol{q}^j \in \mathbb{R}^3$ denotes the joint angle variations of the jth leg, $\mathbf{k}^j \in \mathbb{R}^{3\times3}$ is a diagonal matrix taking the spring constants of the parallel springs installed on the hip, thigh, and calf joints as the diagonal elements.

In the homing pose, the four legs of the quadrupedal robot share the same joint variations and the same Jacobian matrix due to symmetry. That is, $\delta\boldsymbol{q} = \delta\boldsymbol{q}^j$ and $\mathbf{J_0} = \mathbf{J}_0^j$ ($\mathbf{J}_0^j$ represents the contact Jacobian of the jth leg in the homing configuration). In the current design, four legs also share the same parallel spring combination, noted by $\mathbf{k} = \mathbf{k}^j, j \in \{1, \ldots, 4\}$. As a result, (30) is simplified as follows:

$$V_{\text{quad}}^{\text{spring}} = 4\left(\frac{1}{2}(\delta\boldsymbol{q})^{\mathrm{T}}\mathbf{k}(\delta\boldsymbol{q})\right). \tag{31}$$

For a SLIP, given the small variation in the leg length ($\delta\boldsymbol{l} = \mathbf{J}_0\delta\boldsymbol{q}$ in the homing pose), the potential energy ($V_{\text{slip}}^{\text{spring}}$) contributed by the spring tension is computed as follows:

$$V_{\text{slip}}^{\text{spring}} = \frac{1}{2}(\delta\boldsymbol{l})^{\mathrm{T}}\mathbf{k}_{\text{equ}}(\delta\boldsymbol{l}) = \frac{1}{2}(\delta\mathbf{q})^{\mathrm{T}}[\mathbf{J}_0^{\mathrm{T}}\mathbf{k}_{\text{equ}}\mathbf{J}_0](\delta\mathbf{q}) \tag{32}$$

where $\mathbf{k}_{\text{equ}} \in \mathbb{R}^{3\times3}$ is the equivalent spring constant matrix.

To match the SLIP model with the quadrupedal model, we demand $V_{\text{quad}}^{\text{spring}} = V_{\text{slip}}^{\text{spring}}$. As a result, we have the following:

$$\mathbf{k}_{\text{equ}} = 4(\mathbf{J}_0^{-1})^{\mathrm{T}}\mathbf{k}\mathbf{J}_0^{-1}. \tag{33}$$

Then, the spring constants k_s of the actuated SLIP is as follows:

$$k_s = \sqrt{(\mathbf{k}_{\text{equ}(0,0)})^2 + (\mathbf{k}_{\text{equ}(1,1)})^2 + (\mathbf{k}_{\text{equ}(2,2)})^2}. \tag{34}$$

APPENDIX C
FEEDFORWARD TORQUE OF ACTUATED SLIP MODEL

Equation (3o) constrains the joint torque within a feasible range, where the operator $\boldsymbol{\tau}(\cdot)$ computes the commanded torque at the kth knot. Specifically, for the jth leg, we have the following:

$$\boldsymbol{\tau}_{(k)}^j = -(\mathbf{J}_{(k)}^j)^{\mathrm{T}}\mathbf{F}_{(k)}^j \tag{35}$$

where $\mathbf{F}_{(k)}^j \in \mathbb{R}^3$, $\boldsymbol{\tau}_{(k)}^j \in \mathbb{R}^3$, and $\mathbf{J}_{(k)}^j \in \mathbb{R}^{3\times3}$ separately denote the GRF, joint torque, and contact Jacobian.

As discussed in many previous works, including [39], the GRF should be carefully distributed to accomplish desired body movements while obeying feasibility constraints. In this work, to simplify the TO formulation, we propose a heuristic rule for calculating the GRF. That is, the total force is balanced by the GRFs on the four legs, considering the force arm of each leg position w.r.t the body center. Following the definition in Fig. 4, the $\mathbf{F}_{(k)}^j$ is computed as follows:

$$\mathbf{F}_{(k)}^j = \lambda_1\lambda_2 m\mathbf{u}_{(k)} \tag{36}$$

[11] If we express $\mathbf{M}(\boldsymbol{c})\boldsymbol{u}$ in (28) in a specific form, such as the formula with the time-varying rest length, the "actuated" SLIP in [31] and [60] is then obtained. In this sense, we can think of the SLIP variants in other work, including [30], [31], [32], [33], and[34], as the special cases of our model.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

where λ_1 and λ_2 are introduced to distribute the force in the sagittal and lateral plane, determined as follows:

$$\lambda_1=\begin{cases}\frac{\boldsymbol{X}_{x(k)}+x_r}{x_f+x_r} & \text{front legs}\\ \frac{x_f-\boldsymbol{X}_{x(k)}}{x_f+x_r} & \text{rear legs}\end{cases},\lambda_2=\begin{cases}\frac{\boldsymbol{X}_{y(k)}+y_r}{y_l+y_r} & \text{left legs}\\ \frac{y_l-\boldsymbol{X}_{y(k)}}{y_l+y_r} & \text{right legs}\end{cases}. \quad (37)$$

Then, by substituting (36) and (37) into (35), the commanded torques for quadrupedal jumping can be obtained.

REFERENCES

[1] C. D. Santina, M. G. Catalano, A. Bicchi, M. Ang, O. Khatib, and B. Siciliano, "Soft robots," *Ency. Robot.*, vol. 489, pp. R639–R641, 2021.

[2] M. Hutter et al., "Anymal-a highly mobile and dynamic quadrupedal robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 38–44.

[3] "Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion," in *Proc. Adapt. Mobile Robot. World Sci.*, 2012, pp. 483–490.

[4] D. Seidel, M. Hermann, T. Gumpert, F. C. Loeffl, and A. A.-Schäffer, "Using elastically actuated legged robots in rough terrain: Experiments with DLR quadruped bert," in *Proc. IEEE Aerosp. Conf. Proc.*, 2020, pp. 1–8.

[5] A. Spröwitz et al., "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 932–950, 2013.

[6] A. B. -Spröwitz, A. A. Sarvestani, M. Sitti, and M. A. Daley, "Birdbot achieves energy-efficient gait with minimal control using avian-inspired leg clutching," *Sci. Robot.*, vol. 7, no. 64, 2022, Art. no. eabg4055.

[7] F. Bjelonic et al., "Learning-based design and control for quadrupedal robots with parallel-elastic actuators," *IEEE Robot. Autom. Lett.*, vol. 8, no. 3, pp. 1611–1618, Mar. 2023.

[8] P. Arm et al., "Spacebok: A dynamic legged robot for space exploration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 6288–6294.

[9] Q. Nguyen, M. J. Powell, B. Katz, J. D. Carlo, and S. Kim, "Optimized jumping on the MIT cheetah 3 robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 7448–7454.

[10] C. Nguyen, L. Bao, and Q. Nguyen, "Continuous jumping for legged robots on stepping stones via trajectory optimization and model predictive control," in *Proc. IEEE Conf. Decis. Control*, 2022, pp. 93–99.

[11] M. Chignoli and S. Kim, "Online trajectory optimization for dynamic aerial motions of a quadruped robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 7693–7699.

[12] M. Chignoli, S. Morozov, and S. Kim, "Rapid and reliable quadruped motion planning with omnidirectional jumping," in, *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 6621–6627.

[13] G. Bellegarda and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," 2020, *arXiv:2011.07089*.

[14] C. Mastalli et al., "A feasibility-driven approach to control-limited DDP," *Auton. Robots*, vol. 46, no. 8, pp. 985–1005, 2022.

[15] S. H. Jeon, S. Kim, and D. Kim, "Online optimal landing control of the MIT mini cheetah," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 178–184.

[16] K. Ye and K. Karydis, "Evaluation of legged robot landing capability under aggressive linear and angular velocities," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 12184–12190.

[17] F. Roscia, M. Focchi, A. D. Prete, D. G. Caldwell, and C. Semini, "Reactive landing controller for quadruped robots," *IEEE Robot. Autom. Lett.*, vol. 8, no. 11, pp. 7210–7217, Nov. 2023.

[18] H. Kolvenbach, E. Hampp, P. Barton, R. Zenkl, and M. Hutter, "Towards jumping locomotion for quadruped robots on the moon," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2019, pp. 5459–5466.

[19] N. Rudin, H. Kolvenbach, V. Tsounis, and M. Hutter, "Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 317–328, Feb. 2022.

[20] Unitree Robotics, "Unitree robot go1," Accessed: Mar. 18, 2023. [Online]. Available: https://m.unitree.com/products/go1

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

[22] C. Nguyen and Q. Nguyen, "Contact-timing and trajectory optimization for 3D jumping on quadruped robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2022, pp. 11 994–11 999.

[23] S. Gilroy et al., "Autonomous navigation for quadrupedal robots with optimized jumping through constrained obstacles," in *Proc. IEEE Int. Autom. Softw. Eng. Conf.*, 2021, pp. 2132–2139.

[24] Z. Song et al., "An optimal motion planning framework for quadruped jumping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2022, pp. 11 366–11 373.

[25] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1154–1171, Aug. 2021.

[26] Y. Ding, C. Li, and H.-W. Park, "Kinodynamic motion planning for multi-legged robot jumping via mixed-integer convex program," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2020, pp. 3998–4005.

[27] M. J. Pollayil et al., "Planning natural locomotion for articulated soft quadrupeds," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 6593–6599.

[28] R. Blickhan, "The spring mass model for running and hopping," *J. Biomech.*, vol. 22, no. 11–12, pp. 1217–1227, 1989.

[29] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proc. Roy. Soc. B*, vol. 273, no. 1603, pp. 2861–2867, 2006.

[30] S. Rezazadeh et al., "Spring-mass walking with atrias in 3D: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot," in *Proc. Prof. Conf. Dyn. Syst. Control. Syst.*, vol. 57243, 2015, Art. no. V001T04A003.

[31] Y. Liu, P. M. Wensing, D. E. Orin, and Y. F. Zheng, "Dynamic walking in a humanoid robot based on a 3D actuated dual-slip model," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5710–5717.

[32] K. Wang, H. Fei, and P. Kormushev, "Fast online optimization for terrain-blind bipedal robot walking with a decoupled actuated slip model," *Front. Robot. AI*, vol. 9, 2022, Art. no. 812258.

[33] X. Xiong and A. D. Ames, "Coupling reduced order models via feedback control for 3D underactuated bipedal robotic walking," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2018, pp. 67–74.

[34] K. Green, R. L. Hatton, and J. Hurst, "Planning for the unexpected: Explicitly optimizing motions for ground uncertainty in running," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 1445–1451.

[35] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2018, pp. 3821–3828.

[36] D. Lakatos et al., "Dynamic locomotion gaits of a compliantly actuated quadruped with slip-like articulated legs embodied in the mechanical design," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3908–3915, Oct. 2018.

[37] D. Calzolari, C. D. Santina, A. M. Giordano, and A. A.-Schäffer, "Single-leg forward hopping via nonlinear modes," in *Proc. Amer. Control Conf.*, 2022, pp. 506–513.

[38] X. Xiong and A. Ames, "3-D underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2405–2425, Aug. 2022.

[39] J. D. Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2018, pp. 1–9.

[40] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," 2019, *arXiv:1909.06586*.

[41] V. Kurtz, H. Li, P. M. Wensing, and H. Lin, "Mini cheetah, the falling cat: A case study in machine learning and trajectory optimization for robot acrobatics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 4635–4641.

[42] Y. Tang et al., "Towards safe landing of falling quadruped robots using a 3-DOF morphable inertial tail," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 1141–1147.

[43] K. Wang, G. Xin, S. Xin, M. Mistry, S. Vijayakumar, and P. Kormushev, "A unified model with inertia shaping for highly dynamic jumps of legged robots," *Mechatron.*, vol. 95, 2023, Art. no. 103040.

[44] H. Qi et al., "Vertical jump of a humanoid robot with CoP-guided angular momentum control and impact absorption," *IEEE Trans. Robot.*, vol. 39, no. 4, pp. 3154–3166, Aug. 2023.

[45] J. T. Bingham, J. Lee, R. N. Haksar, J. Ueda, and C. K. Liu, "Orienting in mid-air through configuration changes to achieve a rolling landing for reducing impact after a fall," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2014, pp. 3610–3617.

[46] J. Qi et al., "Integrated attitude and landing control for quadruped robots in asteroid landing mission scenarios using reinforcement learning," *Acta Astronaut.*, vol. 204, pp. 599–610, 2023.

[47] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to Humanoid Robotics*, vol. 101. Berlin, Germany: Springer, 2014.

[48] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2013, pp. 5134–5140.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.

[49] H. R. Vejdani, A. Wu, H. Geyer, and J. W. Hurst, "Touch-down angle control for spring-mass walking," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5101–5106.
[50] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi–A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.
[51] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Prog..*, vol. 106, pp. 25–57, 2006.
[52] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.
[53] E. Coumans and Y. Bai, "Pybullet, a Python module for Physics simulation for games, robotics and machine learning," 2016–2019. [Online]. Available: http://pybullet.org
[54] Z. Li et al., "Robust and versatile bipedal jumping control through reinforcement learning," *Robot.: Sci. Syst. XIX*, 2023.
[55] N. G. Tsagarakis, S. Morfey, G. M. Cerda, L. Zhibin, and D. G. Caldwell, "Compliant humanoid COMAN: Optimal joint stiffness tuning for modal frequency control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 673–678.
[56] M. J. Pollayil et al., "Choosing stiffness and damping for optimal impedance planning," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 1281–1300, Apr. 2023.
[57] H. Chen, Z. Hong, S. Yang, P. M. Wensing, and W. Zhang, "Quadruped capturability and push recovery via a switched-systems characterization of dynamic balance," *IEEE Trans. Robot.*, vol. 39, no. 4, pp. 2111–2130, Jun. 2023.
[58] J. Ding, T. L. Lam, L. Ge, J. Pang, and Y. Huang, "Safe and adaptive 3-D locomotion via constrained task-space imitation learning," *IEEE/ASME Trans. Mechatron.*, vol. 28, no. 6, pp. 3029–3040, Dec. 2023.
[59] T. Koolen, T. D. Boer, J. Rebula, A. R. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *Int. J. Robot. Res.*, vol. 31, no. 9, pp. 1094–1113, 2012.
[60] Y. Liu, P. M. Wensing, D. E. Orin, and Y. F. Zheng, "Trajectory generation for dynamic walking in a humanoid over uneven terrain using a 3d-actuated dual-slip model," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2015, pp. 374–380.

Jiatao Ding (Member, IEEE) received the B.Eng. degree and Ph.D. degrees in engineering from Wuhan University, Wuhan, China, in 2014 and 2020, respectively.

From 2018 to 2020, he was a Visiting Ph.D. student with the Italian Institute of Technology, Italy. From 2020 to 2022, he was an Assistant Research Scientist with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, China. He is currently a Postdoctoral Researcher with the Department of Cognitive Robotics, Delft University of Technology, Delft, The Netherlands. His research interests include optimal control and robot learning on legged locomotion.

Vassil Atanassov received the B.Eng. degree (first class Hons.) in mechanical engineering from the University of Glasgow, Glasgow, Scotland, and the master's degree (cum laude) in robotics from the Delft University of Technology, Delft, The Netherlands, in 2021 and 2023, respectively. He is currently working toward the Ph.D. degree in robotics with the Department of Engineering Science, University of Oxford, Oxford, U.K.

His research interests include model-based and data-driven control of legged robots.

Edoardo Panichi received the B.Sc. degree (cum laude) in automation engineering from the University of Bologna, Bologna, Italy, in 2021 and the M.Sc. degree (cum laude) in robotics from Delft University of Technology, Delft, The Netherlands, in 2024 .

He is currently a Robotics Engineer developing robotic solutions for the offshore industry with X-Laboratory, Delft University of Technology, Delft The Netherlands. His research interests include robust control systems and imitation learning approaches for quadruped robotics.

Jens Kober (Senior Member, IEEE) received the Ph.D. degree in engineering from TU Darmstadt, Darmstadt, Germany, in 2012.

He was a Postdoctoral Scholar jointly with CoR-Lab, Bielefeld University, Bielefeld, Germany, and with Honda Research Institute Europe, Offenbach, Germany. He is currently an Associate Professor with TU Delft, Delft, The Netherlands.

Dr. Kober was the recipient of the annually awarded Georges Giralt Ph.D. Award for the best Ph.D. thesis in robotics in Europe, the 2018 IEEE RAS Early Academic Career Award, the 2022 RSS Early Career Award, and was a recipient of an ERC starting grant.

Cosimo Della Santina (Senior Member, IEEE) received the Ph.D. degree (cum laude) in robotics from the University of Pisa, Pisa, Italy, in 2019.

From 2017 to 2019, he was a visiting Ph.D. student and a Postdoc with Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology. He was a Senior Postdoc and a Guest Lecturer with the Department of Informatics, Technical University of Munich, in 2020 and 2021, respectively. He is currently an Assistant Professor with TU Delft, Delft, The Netherlands, and a Research Scientist with German Aerospace Institute (DLR), Munich, Germany. His research interest includes providing motor intelligence to physical systems, focusing on elastic and soft robots.

Dr. Santina was the recipient of several awards, including the euRobotics Georges Giralt Ph.D. Award in 2020 and the IEEE RAS Early Academic Career Award in 2023. He was a recipient of a NWO VENI. He is involved as PI in a number of European and Dutch Projects, he is the coDirector of Delft AI Lab SELF.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 16,2024 at 09:50:53 UTC from IEEE Xplore. Restrictions apply.