Adaptive-Force-Based Control of Dynamic Legged Locomotion Over Uneven Terrain

Mohsen Sombolestan and Quan Nguyen

***Abstract*—Agile-legged robots have proven to be highly effective in navigating and performing tasks in complex and challenging environments, including disaster zones and industrial settings. However, these applications commonly require the capability of carrying heavy loads while maintaining dynamic motion. Therefore, this article presents a novel methodology for incorporating adaptive control into a force-based control system. Recent advancements in the control of quadruped robots show that force control can effectively realize dynamic locomotion over rough terrain. By integrating adaptive control into the force-based controller, our proposed approach can maintain the advantages of the baseline framework while adapting to significant model uncertainties and unknown terrain impact models. Experimental validation was successfully conducted on the Unitree A1 robot. With our approach, the robot can carry heavy loads (up to 50% of its weight) while performing dynamic gaits such as fast trotting and bounding across uneven terrains.**

***Index Terms*—Adaptive control, model predictive control (MPC), quadruped robots, unknown impact model.**

I. Introduction

LEGGED robots have numerous potential uses, from search and rescue operations to autonomous construction. To perform these tasks effectively, the robot must accurately understand the environment it will be operating in. However, due to the complexity of the robot and the environment, the robot's model might contain a significant level of uncertainty and affect the robot's stability, particularly when performing agile movements. To overcome these challenges, there is a need to develop a control framework that can effectively compensate for these uncertainties in real time.

The utilization of convex model predictive control (MPC) with the single rigid body (SRB) model in legged robots [1] has greatly enhanced the real-time implementation of diverse walking gaits. Unlike the balance controller based on quadratic programming [2], MPC offers the capability to perform agile motions like jumping [3], [4] and high-speed bounding [5] for quadruped robots. Additionally, MPC exhibits robustness when traversing rough and uneven terrains. However, it is important to note that MPC assumes perfect knowledge of the dynamic model.

To enhance trajectory tracking in the presence of unknown and changing disturbances, researchers have explored the combination of MPC with adaptive control techniques [6], [7], [8]. Additionally, parameter estimation techniques have been employed to improve the robustness of the control system further [9]. These approaches aim to adapt the controller and estimate system parameters to effectively compensate for uncertainties and disturbances, leading to improved trajectory tracking performance. It is worth noting that all of these studies were conducted using a position-based controller model.

In this work, we tackle the legged robot locomotion issue in real-world scenarios with significant uncertainty. The uncertainty can come from both the robot model and the environment. Since our proposed method is based on a force controller, it retains the advantage of robustness to uneven terrain. Thanks to MPC as our baseline controller, our framework can be extended to different locomotion gaits and trajectories without adjusting the controller parameters. Moreover, in our control system, we effectively manage substantial model uncertainty by utilizing the adaptive controller. By implementing adaptive control, our framework evolves into a versatile solution for mitigating persistent disturbances across various operations and over time. Given the adaptive control's capability to address uncertainties continuously, it provides a practical approach for real-world applications in legged robot autonomy, such as rescue missions, inspections, and logistics. This ability to compensate for persistent disturbances in real-world scenarios eliminates the need for recalibration for various tasks, enabling a thorough online operation. As a result, this represents a key contribution to our work, offering a comprehensive approach for legged robot applications and facilitating movement across diverse terrains with unknown impact models.

A. Related Works

1) Offline Learning: The offline learner can leverage a model-based control approach or learn the control system from scratch. Using a model-based method, researchers mainly target learning the dynamic to improve the controller performance [10]. One example of this approach is integrating deep learning with MPC, in which the proposed model tries to learn the cost or dynamic terms of an MPC [11]. This hybrid method

Manuscript received 7 July 2023; revised 21 December 2023; accepted 20 March 2024. Date of publication 25 March 2024; date of current version 10 April 2024. This paper was recommended for publication by Guest Associate Editor Yan Gu and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant IIS-2133091 and in part by the USC startup fund. *(Corresponding author: Mohsen Sombolestan.)*

The authors are with the Department of Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: somboles@usc.edu; quann@usc.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3381554, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3381554

shows considerable improvement for the aerial robot [12] when learning the dynamic model from experimental data. The major limitation of this method is that it is restricted to the dynamic model learned during the training phase. However, the dynamic model is prone to frequent changes in real-world scenarios due to environmental uncertainties and external disturbances.

There has been growing interest in utilizing reinforcement learning (RL) to train models from scratch to overcome the limitations of previous approaches. The key advantage of RL models is their ability to adapt swiftly to changes in real-world environments due to being trained in diverse environments with varying properties. In the case of quadruped robots, an RL model can directly predict appropriate joint torques for traversing different types of terrain, as demonstrated by Chen et al. [13]. Additionally, by training the model to learn foot positions, Bellegarda et al. [14] enable quadrupeds to run quickly while carrying unknown loads. However, these methods heavily rely on domain randomization during training to generalize to challenging environments. Yang et al. [15] also propose an end-to-end RL method that utilizes proprioceptive states and visual feedback to predict environmental changes.

2) Online Learning: To address inaccuracies in model-based controllers, researchers have explored an alternative approach using online learning, mainly supervised learning methods [16], [17], [18]. In this approach, the focus is on learning disturbances online [19], and in some cases, researchers also aim to learn the dynamics of the system itself [20]. Furthermore, this approach has been successfully applied for online calibration of kinematic parameters in legged robots [21]. In addition, a recent study has developed a Lipschitz network method to bridge the model-reality gap in real time [22]. The online learning method is closely related to adaptive control, and numerous studies have explored combining these two approaches [23]. This combination aims to leverage the advantages of both methods, allowing for dynamic adaptation and continuous learning from real-time data to improve control system performance. Perhaps closest to our work in terms of online adaption is the learning method presented in [24] for legged robots. The authors correct the model behind the controller using a supervised learner while the robot is walking in an unknown environment. The data are collected during the robot's operation to learn a linear residual model that can compensate for system errors. However, in the transition from simulation to experiment, the acceleration estimators make noisy data required for training the model. As a result, the method is only applied to estimate the linear terms since the angular terms data proved to be too noisy to be helpful in the model.

To enhance controller efficiency and performance, autotuning methods, particularly for PID controllers, have gained widespread use [25], [26]. These methods fall into two categories: 1) model-based and 2) model-free. Model-based approaches use system model information, often employing the gradient of the performance criterion to enhance local performance [27]. In contrast, model-free methods, such as Markov chain Monte Carlo [28], Gaussian process [29], [30], and deep neural network [31], approximate gradients or surrogate models to boost performance. However, model-based approaches may fail in real-world scenarios due to imperfect dynamic knowledge, and model-free methods such as Bayesian optimization can be inefficient in high-dimensional parameter tuning. Addressing this, recent works [32], [33] directly obtain the gradient of the loss function with respect to controller parameters and apply it to gradient descent for performance improvement. Autotuning generally requires data or a fixed model for training and, therefore, does not fit well for real time and fast adaptation to significant model uncertainty.

3) Adaptive Control: Adaptive control aims to tune the controller's variables online during deployment [34]. Adaptive control has been applied for manipulation tasks to robotic arms [35], mobile robots [36], [37], [38], and quadruped robots [39], [40]. The conventional model reference adaptive control (MRAC) architecture was initially designed for controlling linear systems in the presence of parametric uncertainties [41], [42]. However, it cannot characterize the input/output performance of the system during the transient phase. To address this limitation and improve the transient performance of adaptive controllers, the L_1 adaptive control offers several advantages over traditional MRAC, such as decoupling adaptation and robustness within a control framework [43]. In addition, incorporating a low-pass filter in adaptation law allows the L_1 adaptive control to provide stability [44] and transient performance [45]. Therefore, the L_1 adaptive control technique guarantees robustness with fast adaptation [46], an essential criterion in dynamic robotics applications. Recently, by integrating L_1 adaptive controller and Bayesian learner, researchers leverage the fast adaption performance of the L_1 adaptive controllers and introduce a safe simultaneous control and learning framework [47], [48].

For legged robots, the adaptive controller has also been employed to find the value and location of the center of mass (COM) [49]. Our work on L_1 adaptive control for bipedal robots [50] considers a control Lyapunov function (CLF)-based controller as a closed-loop nonlinear reference model for the L_1 adaptive controller. It was validated for the robot's walking [51] and running [52]. However, the control framework in this prior work is based on hybrid zero dynamics (HZD) [53], which uses joint position control to track the desired trajectory from optimization for each robot joint. Moreover, in [54], an adaptive control based on a CLF is designed for quadrupeds to interact with unknown objects. Then, they combined the criteria derived by adaptive control as a constraint in an MPC framework. However, adding more inequality constraints to MPC makes the controller more complex in terms of computation. In our approach, we compute a residual vector to compensate for dynamic uncertainty, which makes the controller more time-efficient. Additionally, by employing our method, the robot can adapt to terrains with unknown impact models.

B. Contributions

A preliminary version of this research appeared in [55]; however, this article presents several novel contributions to the prior work. This work incorporates the L_1 adaptive controller into the MPC. The proposed control system leverages MPC due to its robustness to uneven terrain, contact constraints, and

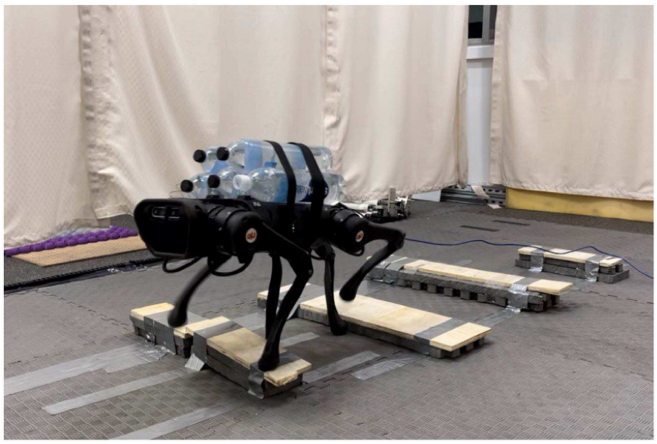


Fig. 1. Our proposed adaptive MPC is successfully validated in an experiment on a Unitree A1 robot while carrying an unknown load of 5 kg (almost 50% of body weight) on rough terrain. Experimental results video: https://youtu.be/5t1mSh0q3lk.

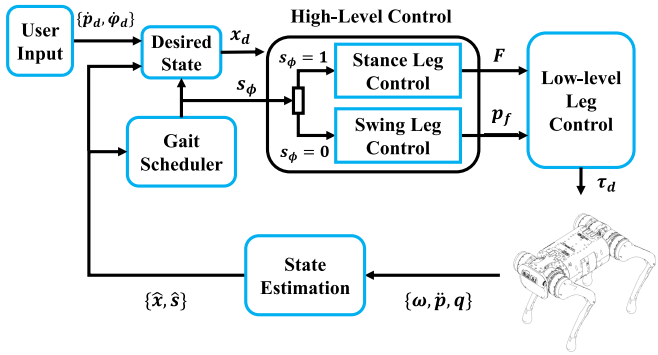


Fig. 2. *Baseline Control Structure.* Block diagram of a control architecture for a quadruped robot. For the stance leg control, we use two common baseline control systems: 1) QP-based balancing controller and 2) MPC.

generalization to different locomotion gaits. Moreover, by integrating adaptive control into MPC, the proposed model can compensate for significant model uncertainty. In the previous work [55], the robot can only perform quasi-static walking; however, in this work, the robot can perform dynamic motions thanks to MPC. Finally, the authors present new hardware experiments to demonstrate the effectiveness of the proposed adaptive MPC (as illustrated in Fig. 1). The main contributions of the article are as follows.

1) We introduce a novel control system that combines the L_1 adaptive control into the force-based control system, designed to address the challenges posed by model uncertainty in real-world applications.
2) Thanks to MPC, our approach offers greater versatility as it can be adapted to a wide range of locomotion gaits and trajectories. Moreover, our method can handle terrain uncertainty, allowing the robot to navigate rough terrains and high-sloped terrain, such as grass and gravel.
3) By integrating the adaptive control into MPC, it is possible for quadruped robots to carry an unknown heavy load (up to 50% of the robot's weight) across challenging terrains, with the capability of executing dynamic gaits such as fast trotting and bounding. This is a significant improvement compared to our previous work, which only allowed the robot to perform quasi-static walking.
4) The combination of using MPC for both the reference model and the real model in the adaptive controller makes the control system computationally expensive, leading to potential delays in computation. To ensure real-time performance, we have developed an update frequency scheme for the control system, which allows for the optimized allocation of processing resources to each control component.
5) Our proposed approach enables the control system to adapt to terrains with unknown impact models, such as soft terrain. Traversing soft terrain is a challenging task for quadruped robots. Using our method, the A1 robot can walk on double-foam terrain in different directions. In comparison, the robot cannot maintain its balance using the baseline controller, resulting in a collapse.

The rest of this article is organized as follows. Section II presents the baseline control architecture for quadruped robots and provides some knowledge on force-based controllers. In Section III, we will briefly present an overview of our control approach. Then, our proposed adaptive force-based controller using balance controller and MPC will be elaborated in Sections IV and V, respectively. Furthermore, the numerical and experimental validation are shown in Section VII. Finally, Section VIII concludes this article.

II. Preliminaries

In this section, we present the background on the control architecture of quadruped robots and describe each control component. According to the work in [56], the robot's control system consists of several modules, including a high-level controller, low-level controller, state estimation, and gait scheduler as presented in Fig. 2.

A reference trajectory can be generated for high-level control from user input and state estimation. The gait scheduler defines the gait timing and sequence to switch between each leg's swing and stance phases. The high-level part controls the position of the swing legs and optimal ground reaction force (GRF) for stance legs based on the user commands and gait timing. As the baseline for the stance leg controller, we will use two common approaches: 1) quadratic program (QP)-based balancing controller [2] and 2) MPC [1]. The low-level leg control converts the command generated by high-level control into joint torques for each motor. These modules of the control architecture will be described briefly in the following sections. More details can be found in [1], [2], and [56].

A. Gait Scheduler

The A1's gait is defined by a finite state machine using a leg-independent phase variable to schedule contact and swing phases for each leg [56]. The gait scheduler utilizes independent Boolean variables to define contact states scheduled $s_\phi \in \{1 =$

contact, $0 =$ swing$\}$ and switch each leg between swing and stance phases. Based on the contact schedule, the controller will execute either position control during swing or force control during stance for each leg.

In our previous work [55], we focused on the application of load-carrying tasks, where the load is unknown to the robot or the control system. Having more legs on the ground during walking could also mean that the robot could produce a more significant total GRF to support the heavy load. Therefore, we used a quasi-static walking gait to maximize the number of legs on the ground during walking (i.e., three stance legs and one swing leg throughout the gait). However, in this article, our framework is not limited by any specific gait. Similar to the baseline MPC control approach [1], the approach can work for different gaits by only changing the gait definition in the gait scheduler.

B. Desired Trajectory

The desired trajectory is generated based on the robot's velocity command. The robot operator commands xy-velocity and yaw rate, and then xy-position and yaw are determined by integrating the corresponding velocity. z position contains a constant value of 0.3 m, and the remaining states (roll, roll rate, pitch, pitch rate, and z-velocity) are always zero.

C. SRB Model of Robot

Due to the complexity of the legged robot, a simplified rigid-body model has been used to present the system's dynamic. This model lets us calculate the GRFs in real time. A few assumptions have been made to achieve simplified robot dynamics [1].

Assumption 1: The robot has low inertia legs, so their effect is negligible on the robot's rigid body dynamic.

Assumption 2: For small values of roll (ϕ) and pitch (θ), the rotation matrix $\boldsymbol{R}$, which transforms from the body to world coordinates, can be approximated as the rotation matrix corresponding to the yaw angle (ψ)

$$\boldsymbol{R} \cong \boldsymbol{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

Therefore, by defining the robot's orientation as a vector of Z–Y–X Euler angles $\boldsymbol{\Theta} = [\phi, \theta, \psi]^T$, the rate of change of the robot's orientation can be approximated as [1]

$$\dot{\boldsymbol{\Theta}} \cong \boldsymbol{R}_z(\psi)\boldsymbol{\omega}_b \tag{2}$$

where $\boldsymbol{\omega}_b$ is the robot's angular velocity in the world frame.

Assumption 3: For small angular velocity, the following approximation can be made:

$$\frac{d}{dt}\left(\boldsymbol{I}_G\boldsymbol{\omega}_b\right) = \boldsymbol{I}_G\dot{\boldsymbol{\omega}}_b + \boldsymbol{\omega}_b \times \left(\boldsymbol{I}_G\boldsymbol{\omega}_b\right) \approx \boldsymbol{I}_G\dot{\boldsymbol{\omega}}_b \tag{3}$$

where $\boldsymbol{I}_G \in \mathbb{R}^{3\times3}$ is the moment of inertia in the world frame.

Based on the earlier assumptions, the state representation of the system is as follows [1]:

$$\begin{bmatrix} \dot{\boldsymbol{p}}_c \\ \dot{\boldsymbol{\Theta}} \\ \ddot{\boldsymbol{p}}_c \\ \dot{\boldsymbol{\omega}}_b \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{1}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{R}_z(\psi) \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 \end{bmatrix}}_{\boldsymbol{D}\in\mathbb{R}^{12\times12}} \underbrace{\begin{bmatrix} \boldsymbol{p}_c \\ \boldsymbol{\Theta} \\ \dot{\boldsymbol{p}}_c \\ \boldsymbol{\omega}_b \end{bmatrix}}_{\boldsymbol{X}\in\mathbb{R}^{12}} + \underbrace{\begin{bmatrix} \boldsymbol{0}_{6\times12} \\ \boldsymbol{M}^{-1}\boldsymbol{A} \end{bmatrix}}_{\boldsymbol{H}\in\mathbb{R}^{12\times12}} \boldsymbol{F} + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ \boldsymbol{G} \end{bmatrix} \tag{4}$$

with

$$\begin{aligned} \boldsymbol{M} &= \begin{bmatrix} m\boldsymbol{1}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{I}_G \end{bmatrix} \in \mathbb{R}^{6\times6} \\ \boldsymbol{A} &= \begin{bmatrix} \boldsymbol{1_3} & \cdots & \boldsymbol{1}_3 \\ [\boldsymbol{p}_1 - \boldsymbol{p}_c]\times & \cdots & [\boldsymbol{p}_4 - \boldsymbol{p}_c]\times \end{bmatrix} \in \mathbb{R}^{6\times12} \\ \boldsymbol{G} &= \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{0}_{3\times1} \end{bmatrix} \in \mathbb{R}^6 \end{aligned} \tag{5}$$

where m is the robot's mass, $\boldsymbol{g} \in \mathbb{R}^3$ is the gravity vector, $\boldsymbol{p}_c \in \mathbb{R}^3$ is the position of the COM, $\boldsymbol{p}_i \in \mathbb{R}^3$ ($i \in \{1, 2, 3, 4\}$) are the positions of the feet, $\ddot{\boldsymbol{p}}_c \in \mathbb{R}^3$ is body's linear acceleration, $\dot{\boldsymbol{\omega}}_b \in \mathbb{R}^3$ is angular acceleration, and $\boldsymbol{F} = [\boldsymbol{F}_1^T, \boldsymbol{F}_2^T, \boldsymbol{F}_3^T, \boldsymbol{F}_4^T]^T \in \mathbb{R}^{12}$ are the GRFs acting on each of the robot's four feet. The term $[\boldsymbol{p}_i - \boldsymbol{p}_c]\times$ is the skew-symmetric matrix representing the cross product $(\boldsymbol{p}_i - \boldsymbol{p}_c) \times \boldsymbol{F}_i$. Note that $\boldsymbol{p}_i$ and $\boldsymbol{F}_i$ are presented in the world frame. Therefore, the state representation of the system can be rewritten in the compact form

$$\dot{\boldsymbol{X}} = \boldsymbol{D}\boldsymbol{X} + \boldsymbol{H}\boldsymbol{F} + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ \boldsymbol{G} \end{bmatrix}. \tag{6}$$

D. Balance Controller

One of the baseline control approaches for calculating GRFs for quadruped robots is the balance controller presented in [2] based on a QP solver. Based on the assumptions presented in Section II-C, the approximated dynamic model between the body acceleration and GRFs is as follows:

$$\underbrace{\begin{bmatrix} \boldsymbol{1_3} & \cdots & \boldsymbol{1}_3 \\ [\boldsymbol{p}_1 - \boldsymbol{p}_c]\times & \cdots & [\boldsymbol{p}_4 - \boldsymbol{p}_c]\times \end{bmatrix}}_{\boldsymbol{A}\in\mathbb{R}^{6\times12}} \boldsymbol{F} = \underbrace{\begin{bmatrix} m(\ddot{\boldsymbol{p}}_c + \boldsymbol{g}) \\ \boldsymbol{I}_G\dot{\boldsymbol{\omega}}_b \end{bmatrix}}_{\boldsymbol{b}\in\mathbb{R}^6} \tag{7}$$

and the vector $\boldsymbol{b}$ in (7) can be rewritten as

$$\boldsymbol{b} = \boldsymbol{M}\left(\begin{bmatrix} \ddot{\boldsymbol{p}}_c \\ \dot{\boldsymbol{\omega}}_b \end{bmatrix} + \boldsymbol{G}\right). \tag{8}$$

Since the model (7) is linear, the controller can naturally be formulated as the following QP problem [57], which can be solved in real time at 1 kHz

$$\begin{aligned} \boldsymbol{F}^* = \underset{\boldsymbol{F}\in\mathbb{R}^{12}}{\operatorname{argmin}} \; & (\boldsymbol{A}\boldsymbol{F} - \boldsymbol{b}_d)^T\boldsymbol{S}(\boldsymbol{A}\boldsymbol{F} - \boldsymbol{b}_d) \\ & + \gamma_1\|\boldsymbol{F}\|^2 + \gamma_2\|\boldsymbol{F} - \boldsymbol{F}^*_{\text{prev}}\|^2 \\ \text{s.t.} \quad & \underline{\boldsymbol{d}} \le \boldsymbol{C}\boldsymbol{F} \le \bar{\boldsymbol{d}} \\ & \boldsymbol{F}^z_{\text{swing}} = 0 \end{aligned} \tag{9}$$

where $\boldsymbol{b}_d$ is the desired dynamics. The idea of designing $\boldsymbol{b}_d$ will be elaborated in Section IV-A. The cost function in (9) includes terms that consider three goals, including 1) driving the COM position and orientation to the desired trajectories, 2) minimizing the force commands, and 3) minimizing the change of the current solution $\boldsymbol{F}^*$ with respect to the solution from the previous time-step, $\boldsymbol{F}^*_{\text{prev}}$. The priority of each goal in the cost function is defined by the weight parameters $\boldsymbol{S} \in \mathbb{R}^{6\times6}$, γ_1, γ_2 respectively.

The constraints in the QP formulation enforce friction constraints, input saturation, and contact constraints. The constraint $\underline{\boldsymbol{d}} \leq \boldsymbol{C}\boldsymbol{F} \leq \bar{\boldsymbol{d}}$ ensures that the optimized forces lie inside the friction pyramid and the normal forces stay within a feasible range. More details can be found in [2]. Besides the friction constraint, we will enforce the force constraints for the swing legs, $\boldsymbol{F}_{\text{swing}} = \boldsymbol{0}$. The swing legs are then kept in the posing position until they switch to the stance phase. More details on swing leg control are provided in Section II-F.

E. SRB-Based Convex MPC

The calculation of GRFs in quadruped robots is often approached through MPC [1]. This method determines the optimal sequence of inputs over a finite-time horizon, taking into account any constraints within the dynamic model. Every time MPC is executed in the control system, only the first computed control input from the MPC cycle is applied. The inputs determined over the finite time horizon are only used for the optimization problem and are not directly applied in the control system.

To have the dynamic equation in the convenient state-space form, gravity should be added to the state. So, the system can represent as

$$\dot{\boldsymbol{X}}^c = \boldsymbol{D}^c\boldsymbol{X}^c + \boldsymbol{H}^c\boldsymbol{F} \tag{10}$$

where

$$\boldsymbol{X}^c = \begin{bmatrix} \boldsymbol{p}_c \\ \boldsymbol{\Theta} \\ \dot{\boldsymbol{p}}_c \\ \boldsymbol{\omega}_b \\ ||\boldsymbol{g}|| \end{bmatrix} \in \mathbb{R}^{13}$$

$$\boldsymbol{D}^c = \begin{bmatrix} \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{1}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{R}_z(\psi) & \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \frac{\boldsymbol{g}}{||\boldsymbol{g}||} \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} & 0 \end{bmatrix} \in \mathbb{R}^{13\times13}$$

$$\boldsymbol{H}^c = \begin{bmatrix} \boldsymbol{0}_{6\times12} \\ \boldsymbol{M}^{-1}\boldsymbol{A} \\ \boldsymbol{0}_{1\times12} \end{bmatrix} \in \mathbb{R}^{13\times12}. \tag{11}$$

We consider a linear MPC problem with horizon length k as follows:

$$\begin{aligned} \min_{\boldsymbol{F}_i} \quad & \sum_{i=0}^{k-1} {\boldsymbol{e}_{i+1}}^T \boldsymbol{Q}_i \boldsymbol{e}_{i+1} + {\boldsymbol{F}_i}^T \boldsymbol{R}_i \boldsymbol{F}_i \\ \text{s.t.} \quad & \boldsymbol{X}^c_{i+1} = \boldsymbol{D}_{t,i}\boldsymbol{X}^c_i + \boldsymbol{H}_{t,i}\boldsymbol{F}_i \\ & \underline{\boldsymbol{d}} \leq \boldsymbol{C}\boldsymbol{F}_i \leq \bar{\boldsymbol{d}} \end{aligned} \tag{12}$$

where $\boldsymbol{F}_i$ is the computed GRFs at time step i, $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$ are diagonal positive semidefinite matrices, and $\boldsymbol{D}_{t,i}$ and $\boldsymbol{H}_{t,i}$ are discrete-time system dynamics matrices. $\boldsymbol{e}_{i+1}$ is the system state error at time step i define as $\boldsymbol{e} = [\boldsymbol{e}_p,\ \dot{\boldsymbol{e}}_p]^T \in \mathbb{R}^{12}$, with

$$\boldsymbol{e}_p = \begin{bmatrix} \boldsymbol{p}_c - \boldsymbol{p}_{c,d} \\ \log(\boldsymbol{R}_d\boldsymbol{R}^T) \end{bmatrix} \in \mathbb{R}^6, \quad \dot{\boldsymbol{e}}_p = \begin{bmatrix} \dot{\boldsymbol{p}}_c - \dot{\boldsymbol{p}}_{c,d} \\ \boldsymbol{\omega}_b - \boldsymbol{\omega}_{b,d} \end{bmatrix} \in \mathbb{R}^6 \tag{13}$$

where $\boldsymbol{p}_{c,d} \in \mathbb{R}^3$ is the desired position of COM, $\dot{\boldsymbol{p}}_{c,d} \in \mathbb{R}^3$ is the desired body's linear velocity, and $\boldsymbol{\omega}_{b,d} \in \mathbb{R}^3$ is the desired body's angular velocity. The desired and actual body orientations are described using rotation matrices $\boldsymbol{R}_d \in \mathbb{R}^{3\times3}$ and $\boldsymbol{R} \in \mathbb{R}^{3\times3}$, respectively. The orientation error is obtained using the exponential map representation of rotations [58], [59], where the $\log(.) : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ is a mapping from a rotation matrix to the associated rotation vector [2]. The constraint $\underline{\boldsymbol{d}} \leq \boldsymbol{C}\boldsymbol{F}_i \leq \bar{\boldsymbol{d}}$ is equivalent to the constraint in (9) at time step i.

F. Swing Leg Control

For the swing legs, the final footstep location for each leg is calculated from the corresponding hip location using a linear combination of Raibert heuristic [60] and a feedback term from the capture point formulation [56], [61]. The final footstep locations ($\boldsymbol{p}_{f,i}$) are projected on an assumed ground plane and are calculated by

$$\boldsymbol{p}_{f,i} = \boldsymbol{p}_{h,i} + \frac{T_{\boldsymbol{c}_\phi}}{2}\dot{\boldsymbol{p}}_{c,d} + \sqrt{\frac{\boldsymbol{z}_0}{||\boldsymbol{g}||}}\left(\dot{\boldsymbol{p}}_c - \dot{\boldsymbol{p}}_{c,d}\right) \tag{14}$$

where $T_{\boldsymbol{c}_\phi}$ is the stance time scheduled, $\boldsymbol{z}_0$ is the height of locomotion, and $\boldsymbol{p}_{h,i} \in \mathbb{R}^3$ is the position of the corresponding hip i. A Beizer curve calculates the desired swing trajectory (including desired position $\boldsymbol{p}_{d,i}$ and velocity $\boldsymbol{v}_{d,i}$) for swing legs which starts from the initial lift-off position $\boldsymbol{p}_{0,i}$ and ends at the final touch-down location $\boldsymbol{p}_{f,i}$.

G. Low-Level Control

The low-level leg control can generate joint torque commands from the high-level controller. For low-level force control, the controller transforms the force vector to the hip frame by rotation matrix $\boldsymbol{R}$. Then, joint torques are calculated as follows:

$$\boldsymbol{\tau}_{\text{stance},i} = -\boldsymbol{J}(\boldsymbol{q}_i)^T\boldsymbol{R}^T\boldsymbol{F}_i \tag{15}$$

where $\boldsymbol{J}(\boldsymbol{q}_i) \in \mathbb{R}^{3\times3}$ is the leg Jacobian matrix and $\boldsymbol{q_i}$ is the joints angle of ith leg.

To track the desired swing trajectory for each foot, a PD controller with a feedforward term is used to compute joint torques [56]

$$\boldsymbol{\tau}_{\text{swing},i} = \boldsymbol{J}(\boldsymbol{q}_i)^T\left[\boldsymbol{K}_{p,p}(\boldsymbol{p}_{d,i} - \boldsymbol{p}_i) + \boldsymbol{K}_{d,p}(\boldsymbol{v}_{d,i} - \boldsymbol{v}_i)\right] \tag{16}$$

where $\boldsymbol{p}_{d,i}$ and $\boldsymbol{v}_{d,i}$ are desired foot position and velocity, respectively, $\boldsymbol{p}_i$ and $\boldsymbol{v}_i$ are actual foot position and velocity in the robot's frame, respectively, and $\boldsymbol{K}_{p,p} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{K}_{d,p} \in \mathbb{R}^{3\times3}$ are the diagonal matrices of the proportional and derivative gains, respectively.

III. Overview of the Proposed Approach

This section will present an overview of our novel control architecture to incorporate adaptive control into the force control framework. While our approach is not limited to any specific adaptive control approach, we decided to use L_1 adaptive control [46], [50] thanks to its advancement in guaranteeing fast adaptation and smooth control signals. Note that our proposed control system is designed for the stance leg control part in the control architecture of the quadruped robot (see Fig. 2).

Our prior work [50] introduced an adaptive control based on HZD [62] for bipedal robots. HZD is a common control approach for bipedal robots since it can handle hybrid and underactuated dynamics associated with this kind of robot. In this article, however, our approach leverages the combination of the adaptive control and force control system, which calculates GRFs to achieve highly dynamic locomotion for quadrupeds [56]. The use of force control in legged robot systems has several key benefits, including increased robustness in the presence of challenging terrains [2] and the ability to accommodate a wide range of dynamic movements [1], such as various types of locomotion gaits. By combining force control with adaptive control strategies that compensate for model uncertainty, achieving an enhanced control system with these advantages is possible.

The overview of our proposed adaptive-force-based control system is presented in Fig. 3(a). By incorporating an L_1 adaptive controller, we aim to design a combined controller. The force-based controller calculates the optimal GRFs for following the desired trajectory. The adaptive controller calculates the residual parameters for compensating the nonlinear model uncertainty $\boldsymbol{\theta}$ in the system dynamic. Therefore, the goal is to adjust adaptive control signal $\boldsymbol{u}_a$ as well as adaptation law to estimate the model uncertainty ($\hat{\boldsymbol{\theta}}$) correctly and make the real model follow the reference model. For the reference model, we employ a similar linear model described in (6), and we will update the reference model in real time using an ODE solver. Moreover, the vector of uncertainties estimation $\hat{\boldsymbol{\theta}}$ typically has high frequency due to fast estimation in the adaptation law. Thus, we employ a low-pass filter to obtain smooth control signals. We use the same swing leg control to appropriately synchronize the reference and real models. This means that we also use the real model's foot position for the reference model.

In the following sections, we will elaborate on integrating two different force-based controls as the baseline controller into the adaptive control. First, in Section IV, we will describe the proposed method using a QP-based balancing controller, as presented in Fig. 3(b). Then, in Section V, we will show how to incorporate MPC into the adaptive controller in detail, as illustrated in Fig. 3(c).

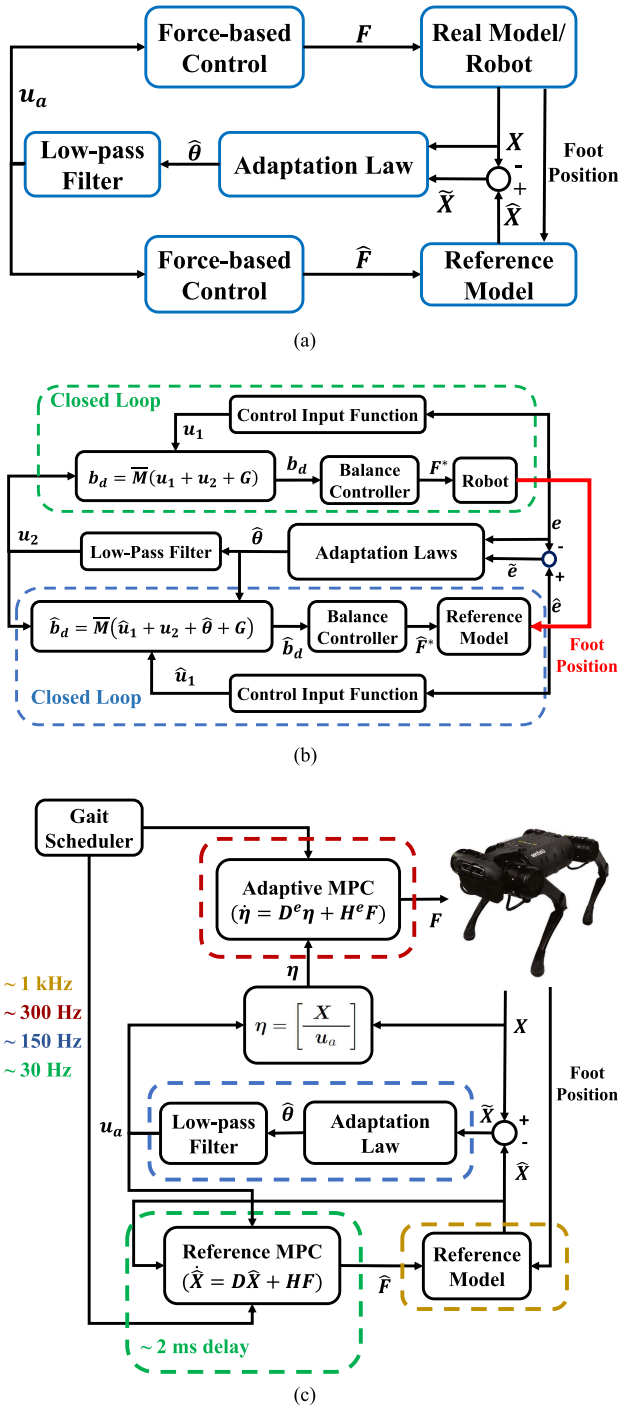


Fig. 3. *Proposed adaptive-force-based control system diagram.* (a) Main structure of the proposed adaptive-force-based control system. (b) Block diagram of the proposed adaptive-QP-based balancing controller. (c) Block diagram of the proposed adaptive MPC. Each dashed line indicates the update frequency for control components.

IV. Adaptive-Force-Based Control Using the Balance Controller

In this section, we use the balance controller as the force-based controller, previously demonstrated in [55]. In Section V, we will present our control framework for integrating the L_1 adaptive control into MPC.

A. Closed-Loop Dynamics

The L_1 adaptive control is designed for trajectory tracking; however, the goal of the balance controller is to compute optimal GRFs. Hence, to integrate the balance controller presented in Section II-D into L_1 adaptive control, we should relate the linear model described in (7) to the closed-loop dynamics.

Let us consider the system state error ($\boldsymbol{e}$) according to (13) as the state variable. Therefore, the closed-loop error dynamics in state-space form can be represented as follows:

$$\dot{\boldsymbol{e}} = \boldsymbol{D}_l \boldsymbol{e} + \boldsymbol{B}\boldsymbol{u} \tag{17}$$

where

$$\boldsymbol{D}_l = \begin{bmatrix} \mathbf{0}_6 & \mathbf{1}_6 \\ \mathbf{0}_6 & \mathbf{0}_6 \end{bmatrix} \in \mathbb{R}^{12\times 12}, \quad \boldsymbol{B} = \begin{bmatrix} \mathbf{0}_6 \\ \mathbf{1}_6 \end{bmatrix} \in \mathbb{R}^{12\times 6} \tag{18}$$

and $\boldsymbol{u} \in \mathbb{R}^6$ is the control input function. By employing a PD control law, we have

$$\boldsymbol{u} = \begin{bmatrix} -\boldsymbol{K}_P & -\boldsymbol{K}_D \end{bmatrix} \boldsymbol{e} \tag{19}$$

where $\boldsymbol{K}_P \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{K}_D \in \mathbb{R}^{6\times 6}$ are diagonal positive-definite matrices. According to definition of matrices $\boldsymbol{D}_l$ and $\boldsymbol{B}$, from (17), it can be obtained that

$$\ddot{\boldsymbol{e}}_p = \begin{bmatrix} \ddot{\boldsymbol{p}}_c - \ddot{\boldsymbol{p}}_{c,d} \\ \dot{\boldsymbol{\omega}}_b - \dot{\boldsymbol{\omega}}_{b,d} \end{bmatrix} = \boldsymbol{u} \tag{20}$$

where $\ddot{\boldsymbol{e}}_p$ is the derivative of $\dot{\boldsymbol{e}}_p$ presented in (13), and $\ddot{\boldsymbol{p}}_{c,d}$ and $\dot{\boldsymbol{\omega}}_{b,d}$ are the desired COM linear acceleration and the desired angular acceleration, respectively. Since the desired trajectory is obtained from the velocity command, both desired accelerations $\ddot{\boldsymbol{p}}_{c,d}$ and $\dot{\boldsymbol{\omega}}_{b,d}$ are zero vectors. Then, from (8) and (20), the desired dynamics can be given by

$$\boldsymbol{b}_d = \boldsymbol{M}(\boldsymbol{u} + \boldsymbol{G}) \tag{21}$$

where $\boldsymbol{M}$ and $\boldsymbol{G}$ are defined in (5). By substituting (21) into the QP problem (9), we can obtain the optimal GRFs as the input for the low-level leg controller. The objective of the QP formulation in (9) is to find a solution that ensures the actual dynamics $\boldsymbol{AF}$ match the desired dynamics $\boldsymbol{b}_d$. The QP-based balance controller can generally achieve the desired control input function outlined in (19), thus keeping the error $\boldsymbol{e}$ within a certain range. However, if the desired dynamics vector $\boldsymbol{b}_d$ violates any of the inequality constraints, such as force limits or friction constraints, the controller may yield an optimal solution $\boldsymbol{F}^*$ that may not completely align with the desired dynamics. With this solution, the optimal dynamic ${\boldsymbol{b}_d}^*$ and $\boldsymbol{u}^*$ can be written as

$${\boldsymbol{b}_d}^* = \boldsymbol{AF}^* \tag{22}$$

$$\boldsymbol{u}^* = \boldsymbol{M}^{-1}\,{\boldsymbol{b}_d}^* - \boldsymbol{G} \tag{23}$$

where in the Appendix, we will show that the $\boldsymbol{u}^*$ remains within a bounded range.

Note that the optimal GRF $\boldsymbol{F}^*$ serves as the control input for the robot, and the variable $\boldsymbol{u}^*$ acts as an input for the closed-loop dynamic. The closed-loop structure for the robot is depicted in Fig. 3(b) (the green dashed line).

B. Effects of Uncertainty on Dynamic

If we consider uncertainty in the dynamic equation (6) and assume that the matrices $\boldsymbol{D}$ and $\boldsymbol{H}$ are not accurate, then we need to present the dynamic based on the nominal matrices $\bar{\boldsymbol{D}}$, $\bar{\boldsymbol{H}}$. The model uncertainty mostly comes from inaccurate values for mass, inertia, and foot position with respect to the COM. In addition, various terrains (e.g., rough terrain or soft terrain) might have different impacts on the robot, which is unknown in a practical situation. Therefore, terrain uncertainty should also be considered in the dynamic model. In this section, we solely derive our control equations based on the model uncertainty. In Section VI, we will elaborate on how our proposed control system can also consider terrain uncertainty.

Another parameter is involved in the dynamic equation, namely the yaw angle. This angle is obtained through the state estimation, and we assumed that the state estimation has minimal uncertainty. According to the definition of matrices $\boldsymbol{D}$ and $\boldsymbol{H}$ in (4), the inaccurate value of the dynamic parameter mentioned earlier reflects on the $\boldsymbol{H}$ matrix. Therefore, the dynamic equation in the presence of uncertainty can be represented as

$$\dot{\boldsymbol{X}} = \boldsymbol{D}\boldsymbol{X} + \left(\bar{\boldsymbol{H}} + \tilde{\boldsymbol{H}}\right)\boldsymbol{F} + \begin{bmatrix} \mathbf{0}_{6\times 1} \\ \boldsymbol{G} \end{bmatrix} \tag{24}$$

where $\tilde{\boldsymbol{H}}$ represent the uncertainty in matrix $\boldsymbol{H}$. It is worth noting that according to the definition of $\boldsymbol{H}$ in (11), the first six rows of $\boldsymbol{H}$ consist of zeros. Thus, we can rephrase the dynamic equation (24) as follows:

$$\dot{\boldsymbol{X}} = \boldsymbol{D}\boldsymbol{X} + \bar{\boldsymbol{H}}\boldsymbol{F} + \boldsymbol{B}\boldsymbol{G} + \boldsymbol{B}\boldsymbol{\theta} \tag{25}$$

where $\boldsymbol{\theta} \in \mathbb{R}^6$ is the vector of uncertainty for six corresponding equations and is defined as follows:

$$\boldsymbol{\theta} \triangleq \boldsymbol{B}^T \tilde{\boldsymbol{H}} \boldsymbol{F}. \tag{26}$$

With reference to the state representation given by (25), the vector $\boldsymbol{\theta}$ can be interpreted as a time-varying disturbance affecting the body and orientation accelerations.

The uncertainty vector $\boldsymbol{\theta}$ depends on both time t and $\boldsymbol{F}$. Since $\boldsymbol{F}$ is obtained through the QP problem (9), it is a function of $\boldsymbol{b}_d$. Furthermore, $\boldsymbol{b}_d$ is a function of $\boldsymbol{u}$ according to (21). Considering that $\boldsymbol{u}$ is determined by the PD control (19), we can conclude that $\boldsymbol{\theta}$ is a function of both the tracking error $\boldsymbol{e}$ and time t. As a result, for any given time t, it is always possible to find $\boldsymbol{\alpha}(t) \in \mathbb{R}^6$ and $\boldsymbol{\beta}(t) \in \mathbb{R}^6$ satisfying [43]

$$\boldsymbol{\theta}(\boldsymbol{e}, t) = \boldsymbol{\alpha}(t)||\boldsymbol{e}|| + \boldsymbol{\beta}(t). \tag{27}$$

C. Designing Adaptive Controller for Compensating the Uncertainty

By incorporating L_1 adaptive controller, we want to design a combined controller $\boldsymbol{u} = \boldsymbol{u}_1 + \boldsymbol{u}_2$, where $\boldsymbol{u}_1$ is the control input to follow the desired trajectory for the nominal model as presented in (19) and $\boldsymbol{u}_2$ is to compensate the nonlinear model uncertainties $\boldsymbol{\theta}$. Therefore, the goal is to adjust the control signal $\boldsymbol{u}_2$ so that the real model can follow the reference model. For the reference model, we employ a similar linear model described in (7) which, instead of $\boldsymbol{M}$, the nominal matrix $\bar{\boldsymbol{M}}$ is being used. The proposed force-based adaptive control diagram based on a balance controller is presented in Fig. 3(b).

The duplicate version of (25) for state-space representation presented in (17) by considering combined controller $\boldsymbol{u} = \boldsymbol{u}_1 + \boldsymbol{u}_2$ is as follows:

$$\dot{\boldsymbol{e}} = \boldsymbol{D}_l \boldsymbol{e} + \boldsymbol{B}\boldsymbol{u}_1 + \boldsymbol{B}\left(\boldsymbol{u}_2 + \boldsymbol{\theta}\right). \tag{28}$$

Note that the vector of uncertainty $\boldsymbol{\theta}$ in (25) and (28) are not the same since the state vector of (25) is $\boldsymbol{X}$. In contrast, the state vector of (28) is system error $\boldsymbol{e}$.

The state representation for the reference model can be expressed as follows:

$$\dot{\hat{\boldsymbol{e}}} = \boldsymbol{D}_l \hat{\boldsymbol{e}} + \boldsymbol{B}\hat{\boldsymbol{u}}_1 + \boldsymbol{B}(\boldsymbol{u}_2 + \hat{\boldsymbol{\theta}}) \tag{29}$$

where

$$\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\alpha}}||\boldsymbol{e}|| + \hat{\boldsymbol{\beta}} \tag{30}$$

and $\hat{\boldsymbol{u}}_1$ is defined as

$$\hat{\boldsymbol{u}}_1 = \begin{bmatrix} -\boldsymbol{K}_P & -\boldsymbol{K}_D \end{bmatrix} \hat{\boldsymbol{e}}. \tag{31}$$

To compensate the estimated uncertainty $\hat{\boldsymbol{\theta}}$, we can just simply choose $\boldsymbol{u}_2 = -\hat{\boldsymbol{\theta}}$ to obtain

$$\dot{\hat{\boldsymbol{e}}} = \boldsymbol{D}_l \hat{\boldsymbol{e}} + \boldsymbol{B}\hat{\boldsymbol{u}}_1. \tag{32}$$

However, $\hat{\boldsymbol{\theta}}$ typically has high frequency due to fast estimation in the adaptation law. Therefore, we employ a low-pass filter to obtain smooth control signals as follows:

$$\boldsymbol{u}_2 = -C(s)\hat{\boldsymbol{\theta}} \tag{33}$$

where $C(s)$ is a second-order low-pass filter with a magnitude of 1

$$C(s) = \frac{{\omega_n}^2}{s^2 + 2\zeta\omega_n s + {\omega_n}^2}. \tag{34}$$

According to (21), $\boldsymbol{b}_d$ for the real model in the presence of uncertainty get the following form:

$$\boldsymbol{b}_d = \bar{\boldsymbol{M}}(\boldsymbol{u}_1 + \boldsymbol{u}_2 + \boldsymbol{G}). \tag{35}$$

Respectively, $\hat{\boldsymbol{b}}_d$ for reference model is as follows:

$$\hat{\boldsymbol{b}}_d = \bar{\boldsymbol{M}}\left(\hat{\boldsymbol{u}}_1 + \boldsymbol{u}_2 + \hat{\boldsymbol{\theta}} + \boldsymbol{G}\right). \tag{36}$$

The QP solver outlined in (9) allows us to obtain the optimal GRFs for the real model. Similarly, the optimal GRFs $\hat{\boldsymbol{F}}$ for the reference model can be obtained as follows:

$$\begin{aligned} \hat{\boldsymbol{F}}^* = \underset{\hat{\boldsymbol{F}} \in \mathbb{R}^{12}}{\operatorname{argmin}} \; & \left(\hat{\boldsymbol{A}}\hat{\boldsymbol{F}} - \hat{\boldsymbol{b}}_d\right)^T \boldsymbol{S}\left(\hat{\boldsymbol{A}}\hat{\boldsymbol{F}} - \hat{\boldsymbol{b}}_d\right) \\ & + \gamma_1 \|\hat{\boldsymbol{F}}\|^2 + \gamma_2 \|\hat{\boldsymbol{F}} - \hat{\boldsymbol{F}}^*_{\text{prev}}\|^2 \\ \text{s.t.} \quad & \underline{\boldsymbol{d}} \le \boldsymbol{C}\hat{\boldsymbol{F}} \le \bar{\boldsymbol{d}} \\ & \hat{\boldsymbol{F}}^z_{\text{swing}} = 0. \end{aligned} \tag{37}$$

Define the difference between the real model and the reference model $\tilde{\boldsymbol{e}} = \hat{\boldsymbol{e}} - \boldsymbol{e}$, we then have

$$\dot{\tilde{\boldsymbol{e}}} = \boldsymbol{D}_l \tilde{\boldsymbol{e}} + \boldsymbol{B}\tilde{\boldsymbol{u}}_1 + \boldsymbol{B}\left(\tilde{\boldsymbol{\alpha}}||\boldsymbol{e}|| + \tilde{\boldsymbol{\beta}}\right) \tag{38}$$

where

$$\tilde{\boldsymbol{u}}_1 = \hat{\boldsymbol{u}}_1 - \boldsymbol{u}_1,\ \tilde{\boldsymbol{\alpha}} = \hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha},\ \tilde{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}} - \boldsymbol{\beta}. \tag{39}$$

As a result, we will estimate $\boldsymbol{\theta}$ indirectly through $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, or the values of $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ computed by the following adaptation laws based on the projection operators [63]:

$$\dot{\hat{\boldsymbol{\alpha}}} = \boldsymbol{\Gamma}\text{Proj}\left(\hat{\boldsymbol{\alpha}}, \boldsymbol{y}_\alpha\right),\ \dot{\hat{\boldsymbol{\beta}}} = \boldsymbol{\Gamma}\text{Proj}\left(\hat{\boldsymbol{\beta}}, \boldsymbol{y}_\beta\right) \tag{40}$$

where $\boldsymbol{\Gamma} \in \mathbb{R}^{6\times 6}$ is a symmetric positive-definite matrix. The projection functions $\boldsymbol{y}_\alpha \in \mathbb{R}^6$ and $\boldsymbol{y}_\beta \in \mathbb{R}^6$ are

$$\begin{aligned} \boldsymbol{y}_\alpha &= -\boldsymbol{B}^T\boldsymbol{P}\tilde{\boldsymbol{e}}||\boldsymbol{e}|| \\ \boldsymbol{y}_\beta &= -\boldsymbol{B}^T\boldsymbol{P}\tilde{\boldsymbol{e}} \end{aligned} \tag{41}$$

where $\boldsymbol{P} \in \mathbb{R}^{12\times 12}$ is a positive-definite matrix that is defined according to the stability criteria using the Lyapunov equation. Moreover, the system's stability proof is provided in the Appendix.

V. Adaptive-Force-Based Control Using MPC

MPC has been widely used across various fields, from finance to robotics. One of MPCs main advantages is its ability to handle complex systems with multiple inputs and outputs while considering hard control constraints [64]. MPC has also been applied to quadruped robots, providing stable locomotion [1]. Thanks to dynamic prediction in MPC, using the same control framework can achieve different dynamic locomotion gaits. However, MPCs limitations become evident when dealing with significant uncertainty in the dynamic model. For instance, in the case of a quadruped robot carrying an unknown heavy load, MPC fails to track the desired state trajectory, resulting in unstable behavior and deviation from the desired trajectory, especially with dynamic gaits like bounding. Furthermore, the ability of a robot to traverse soft terrain where the impact model is unknown can present a significant challenge. Our proposed approach can tackle this challenge effectively, and we will discuss how it handles the terrain unknown impact model in Section VI.

In Section IV, we presented an adaptive-force-based control framework based on the balance controller. The balance controller relies on a QP solver, which is simple to put into practice and well-suited for slow and safe motions such as standing and quasi-static walking. Additionally, the balance controller is an instantaneous control technique, meaning it does not predict the robot's future movement. As a result, the balance controller proves to be ineffective in fast-paced, highly dynamic scenarios. On the other hand, MPC has shown great potential in handling agile motions, even when it comes to underactuated gaits such as bounding.

In this section, we will present a novel control architecture to integrate adaptive control into the MPC framework. By this proposed framework, we can achieve fast and robust locomotion in the presence of uncertainties. This framework can also be extended to accommodate various dynamic gaits in legged robots, such as trotting and bounding. As discussed in a previous section, our approach is not restricted to a specific type of adaptive control. Still, we have chosen to utilize L_1 adaptive control, which has demonstrated advantages over other adaptive control techniques. The first step in integrating L_1 adaptive control and MPC is understanding the importance of a reference model and the challenges in synchronizing the real and reference models. We then present our proposed adaptive MPC, which combines

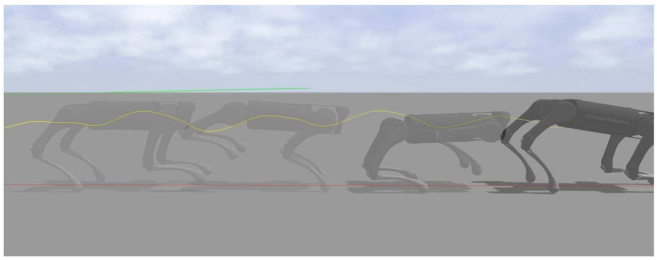


Fig. 4. *Motion snapshot of the robot with bounding gaits.* A simple controller cannot easily predict the quadruped's COM motion (yellow line). This illustration can represent the importance of using MPC for the reference model.

conventional MPC [1] with adaptive control. Finally, we address the challenge of real-time computation while having two MPCs in our control system. We will elaborate on how to adjust the frequency of each control component in an optimized manner to allocate enough computation resources for critical control parts and achieve real-time computation.

A. Reference Model

Our method aims to design a combined controller based on MPC and L_1 adaptive control that the real model follows the reference model. In accordance with our previous discussion in Section IV-C, the combined controller incorporates a control signal $\boldsymbol{u}_2$ to account for model uncertainty, as indicated in (28). In this section, the auxiliary control signal for this purpose is $\boldsymbol{u}_a \in \mathbb{R}^6$; thus, the uncertain dynamic equation (25) can be rewritten as follows:

$$\dot{\boldsymbol{X}} = \boldsymbol{D}\boldsymbol{X} + \bar{\boldsymbol{H}}\boldsymbol{F} + \boldsymbol{B}\boldsymbol{G} + \boldsymbol{B}(\boldsymbol{u}_a + \boldsymbol{\theta}). \tag{42}$$

The reference model is similar to the quasi-linear model described in (6) which, instead of $\boldsymbol{H}$, the nominal matrix $\bar{\boldsymbol{H}}$ is being used. The proposed adaptive MPC diagram is presented in Fig. 3(c).

We consider a reference model for L_1 adaptive control that arises from MPC. The MPC method is computationally expensive, but replacing it with other simpler control methods, such as the balance controller, while simulating the robot's performance using dynamic gaits such as bounding is impossible. The reason is that in bounding gait, the robot's two feet on either the front or rear side touch the ground at each time step, making it challenging to accurately control the height and pitch angle. The MPC approach balances the error in the height and pitch angle and, based on the predicted dynamics of the system in the future, computes the optimal GRFs. As seen in Fig. 4, the COM height oscillates around the desired value. Thus, the underactuated nature of certain gaits, such as bounding, necessitates the use of MPC as the control system for the reference model.

When implementing MPC for a reference model, one challenge is ensuring that the reference model is synchronized with the real model. This is particularly important when the robot performs a gait with a periodic behavior, such as bounding (see Fig. 4). In order to correctly compare the real model with the reference model, both should have the same gait schedule. Additionally, the adaptive MPC proposed for legs in the stance phase is independent of the swing leg control. However, the foot position is crucial in calculating the moment of GRF around the COM. Therefore, to maintain consistency between the real and reference models, it is important to ensure that the real robot's foot position is fed into the reference model, as shown in Fig. 3(c).

The reference model can be expressed as follows:

$$\dot{\hat{\boldsymbol{X}}} = \boldsymbol{D}\hat{\boldsymbol{X}} + \bar{\boldsymbol{H}}\hat{\boldsymbol{F}} + \boldsymbol{B}\boldsymbol{G} + \boldsymbol{B}\left(\boldsymbol{u}_a + \hat{\boldsymbol{\theta}}\right) \tag{43}$$

where

$$\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\alpha}}||\boldsymbol{e}|| + \hat{\boldsymbol{\beta}}. \tag{44}$$

In this case, similar to Section IV, we use a second-order low-pass filter, same as (34). Therefore, the auxiliary control signal would be

$$\boldsymbol{u}_a = -C(s)\hat{\boldsymbol{\theta}}. \tag{45}$$

By defining the difference between the real model and the reference model $\tilde{\boldsymbol{X}} = \hat{\boldsymbol{X}} - \boldsymbol{X}$, we then have

$$\dot{\tilde{\boldsymbol{X}}} = \boldsymbol{D}\tilde{\boldsymbol{X}} + \bar{\boldsymbol{H}}\tilde{\boldsymbol{F}} + \boldsymbol{B}\left(\tilde{\boldsymbol{\alpha}}||\boldsymbol{e}|| + \tilde{\boldsymbol{\beta}}\right) \tag{46}$$

where

$$\tilde{\boldsymbol{F}} = \hat{\boldsymbol{F}} - \boldsymbol{F},\ \tilde{\boldsymbol{\alpha}} = \hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha},\ \tilde{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}} - \boldsymbol{\beta}. \tag{47}$$

Since the desired trajectory for both the real model and the reference model is the same ($\boldsymbol{X}_d = \hat{\boldsymbol{X}}_d$), the difference between the real model and reference model can be defined as

$$\tilde{\boldsymbol{X}} = (\hat{\boldsymbol{X}} - \hat{\boldsymbol{X}}_d) - (\boldsymbol{X} - \boldsymbol{X}_d) = \hat{\boldsymbol{e}} - \boldsymbol{e} = \tilde{\boldsymbol{e}}. \tag{48}$$

Therefore, (46) is equal to the following equation:

$$\dot{\tilde{\boldsymbol{e}}} = \boldsymbol{D}\tilde{\boldsymbol{e}} + \bar{\boldsymbol{H}}\tilde{\boldsymbol{F}} + \boldsymbol{B}\left(\tilde{\boldsymbol{\alpha}}||\boldsymbol{e}|| + \tilde{\boldsymbol{\beta}}\right). \tag{49}$$

The adaption laws and projection functions for computing the value of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the same as (40) and (41), respectively. Moreover, the stability of the control system can be proven using the same logic provided in the Appendix.

B. Adaptive MPC

After computing the auxiliary control signal $\boldsymbol{u}_a$ using the adaptive controller presented in the previous section, we will integrate the $\boldsymbol{u}_a$ with the conventional MPC for legged locomotion [1] and propose our adaptive MPC framework. We treat the auxiliary control signal $\boldsymbol{u}_a$ as a residual vector in the system's equation to compensate for dynamic uncertainty. Therefore, $\boldsymbol{u}_a$ should be combined into the state vector and (42) can be written as follows:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{D}^e\boldsymbol{\eta} + \bar{\boldsymbol{H}}^e\boldsymbol{F} + \boldsymbol{B}^e\boldsymbol{\theta} \tag{50}$$

with the following extended matrices:

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{X}^c \\ \hline \boldsymbol{u}_a \end{bmatrix} \in \mathbb{R}^{19}$$

$$\boldsymbol{D}^e = \left[\begin{array}{c|c} \boldsymbol{D}^c_{13\times 13} & \begin{matrix}\boldsymbol{0}_{6\times 6} \\ \boldsymbol{1}_{6\times 6} \\ \boldsymbol{0}_{1\times 6}\end{matrix} \\ \hline \boldsymbol{0}_{6\times 13} & \boldsymbol{0}_{6\times 6} \end{array}\right] \in \mathbb{R}^{19\times 19}$$

$$\bar{\boldsymbol{H}}^e = \begin{bmatrix} \bar{\boldsymbol{H}}^c \\ \hline \boldsymbol{0}_{6\times 12} \end{bmatrix} \in \mathbb{R}^{19\times 12}$$
$$\boldsymbol{B}^e = \begin{bmatrix} \boldsymbol{B} \\ \hline \boldsymbol{0}_{7\times 6} \end{bmatrix} \in \mathbb{R}^{19\times 6} \tag{51}$$

where $\bar{\boldsymbol{H}}^c$ is the nominal value of $\boldsymbol{H}^c$. The definition of $\boldsymbol{X}^c$, $\boldsymbol{D}^c$, and $\boldsymbol{H}^c$ can be found in (11). Although $\boldsymbol{u}_a$ is considered a part of the state vector in (50), it is just a residual vector for compensating dynamic uncertainty. Therefore, $\boldsymbol{u}_a$ is constant in the state space equation and over the horizons. To this end, the components associated with $\boldsymbol{u}_a$ in matrices $\boldsymbol{D}^e$ and $\bar{\boldsymbol{H}}^e$ are assigned zero, which means $\dot{\boldsymbol{u}}_a = 0$. Note that the value of $\boldsymbol{u}_a$ will be updated according to the adaptive law, but it is constant during the prediction horizons.

The state representation in (50) is also convenient for discretization methods such as zero-order hold [65] for MPC. Therefore, our adaptive MPC can be designed according to (12) and based on the following discrete-time dynamic:

$$\boldsymbol{\eta}_{i+1} = \boldsymbol{D^e}_{t,i}\boldsymbol{\eta}_{t,i} + \bar{\boldsymbol{H}}^e_{t,i}\boldsymbol{F}_i. \tag{52}$$

C. Real-Time Computation

The main challenge in executing our proposed adaptive MPC framework is ensuring that the computation required is fast enough for hardware experiments in real time. If the controller is unable to perform updates at a high frequency, it could result in the robot collapsing during dynamic motion. The control system comprises two MPCs, each with 13–19 states predicted over 10 horizons. To ensure the robot's balance and allocate sufficient computation resources to each control component, we have devised a scheme, as depicted in Fig. 3(c), to update each control component in an optimized manner.

The robot's sensory data update in real time with a frequency of 1 kHz. Thus, the reference model should update with the same frequency to compare the reference model states ($\hat{\boldsymbol{X}}$) and real model states ($\boldsymbol{X}$) correctly. The yellow dashed line in Fig. 3(c) indicates the update frequency for the reference model. We use the *odeint* package from Boost software in C++ [66] to solve the ODE problem associated with the dynamic equation for the reference model.

One of the critical components in our proposed framework is the adaptive MPC, which is responsible for computing the GRF for the robot, as shown in Fig. 3(c). Through our experimentation, we have determined that for robust locomotion with dynamic gaits, the optimal update frequency for the adaptive MPC should be 300 Hz. In contrast, the reference MPC, which plays a supporting role in the control system, is less sensitive and runs at a slower rate of 30 Hz. In addition, there is a 2-ms delay between the running of the adaptive MPC and reference MPC to ensure sufficient computational resources are allocated to each component. This means the two MPC frameworks do not run simultaneously in our control system.

VI. Adaptation to Unknown Impact Model

The dynamic formulation presented in Sections IV and V considers the presence of model uncertainty in real-world situations. It is assumed that the terrain is hard enough to allow the robot to receive the desired force as GRFs on its feet. However, this assumption may not hold if the robot walks on soft or elastic terrain with an unknown impact model, which may not generate the desired force needed for stable locomotion. Some previous studies have included terrain knowledge and contact models in their balancing controllers to address the soft terrain challenge, mainly using a spring–damper model to characterize the soft terrain [67], [68]. Some control frameworks for adapting to soft terrain in real time have also been developed using iterative learning [69] and whole-body control [70], without prior knowledge about the terrain. This section demonstrates that the proposed method in Sections IV and V can also handle unknown impact models from terrain, allowing the robot to maintain stability while walking on soft terrains.

Equation (15), representing the force-to-torque mapping, holds under the condition that the movement of each leg can be considered negligible. This assumption is reasonable for the stance leg on solid ground. However, when dealing with soft terrain, this mapping is not accurate. The dynamic equation for each leg is expressed as follows:

$$\boldsymbol{\tau}_{\text{stance},i} = \boldsymbol{M}_i(\boldsymbol{q}_i)\ddot{\boldsymbol{q}}_i + \boldsymbol{n}_i(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i) - \boldsymbol{J}(\boldsymbol{q}_i)^T\boldsymbol{R}^T\boldsymbol{F}_i \tag{53}$$

where $\boldsymbol{M}_i(\boldsymbol{q}_i)$ is the inertia matrix, $\boldsymbol{n}_i(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i)$ is the nonlinear term, and $\ddot{\boldsymbol{q}}_i$ is joints acceleration for the ith leg. Remember that Assumption 1 was considered for the rigid body dynamic equation of the robot. However, when addressing the dynamic equation of each leg, it is not valid to neglect the inertia associated with each leg. Furthermore, calculating the joint acceleration poses challenges due to considerable noise, making the complete computation of (53) difficult. This is where adaptive controllers prove to be beneficial.

Assume the computed force $\boldsymbol{F}$ by MPC in (25) cannot be achieved perfectly due to walking on soft terrain. Therefore, (25) can be rewritten as follows:

$$\dot{\boldsymbol{X}} = \boldsymbol{D}\boldsymbol{X} + \bar{\boldsymbol{H}}\left(\boldsymbol{F}_a + \tilde{\boldsymbol{F}}_a\right) + \boldsymbol{B}\boldsymbol{G} + \boldsymbol{B}\boldsymbol{\theta} \tag{54}$$

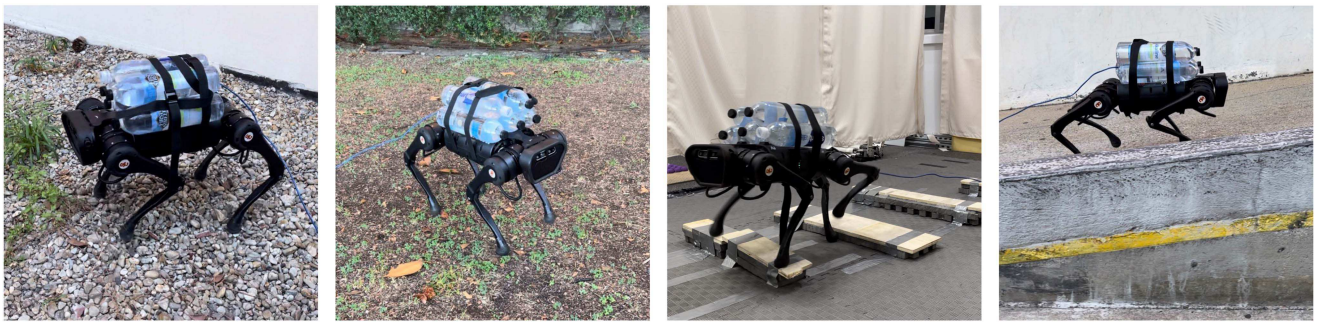
where $\boldsymbol{F}_a$ is the actual GRF applied to the robot and $\tilde{\boldsymbol{F}}_a$ is the difference between the desired GRF and actual reaction force. Therefore, we can reformulate (54) as follows:

$$\dot{\boldsymbol{X}} = \boldsymbol{D}\boldsymbol{X} + \bar{\boldsymbol{H}}\boldsymbol{F}_a + \boldsymbol{B}\boldsymbol{G} + \boldsymbol{B}(\boldsymbol{\theta} + \boldsymbol{\theta}_F). \tag{55}$$

where the uncertainty vector $\boldsymbol{\theta}_F$ is defined as follows:

$$\boldsymbol{\theta}_F \triangleq \boldsymbol{B}^T\bar{\boldsymbol{H}}\tilde{\boldsymbol{F}}_a \tag{56}$$

Equation (55) is in the form of (25), which uses actual GRF instead of the desired GRF. Therefore, all formulations for implementing adaptive controllers are also valid for a situation with an unknown impact model.



(a) (b) (c) (d)

Fig. 5. *Navigating different terrains using our proposed adaptive MPC while carrying an unknown heavy load.* (a) Gravel. (b) Grass. (c) Rough terrain. (d) High-sloped terrain.

TABLE I
CONTROLLER SETTING

Parameter	Value
$\boldsymbol{Q}$	$\text{diag}(2.5, 2.5, 20, 0.25, 0.25, 1.5, 0.2, 0.2, 0.2, 0.1, 0.1, 0.3)$
$\boldsymbol{R}$	$\gamma_1 \boldsymbol{I}_{12}$
$\boldsymbol{\Gamma}$	$\text{diag}(1, 1, 5, 2, 5, 1) \times 10^3$
ω_n	60
ζ	0.7

VII. RESULTS

In this section, we validate our control approach in simulation and hardware experiments on a Unitree A1 robot. All the hardware experiments' computations run on a single PC (Intel i7-6500 U, 2.5 GHz, 64 b). For simulation, the control system is implemented using ROS Noetic in the Gazebo 11 simulator, which provides a high-fidelity simulation of the A1 robot. A video showcasing the results accompanies this article.[1]

We set the control parameters for MPC, the adaption law, and the low-pass filter as presented in Table I. We use one set of parameters for all the experiments with different locomotion gaits, indicating that our approach is easily generalizable. The following sections will introduce different experiment results in terms of model and environment uncertainty (see Fig. 5). In each experiment, the robot starts by using a balance controller to stand up and then switches to the MPC framework for walking or running.

A. Comparative Analysis

In order to evaluate the performance of our proposed adaptive MPC method, we conduct a comparative experiment with the conventional MPC method presented in [1]. The objective is to understand the advantages of integrating the adaptive controller into MPC for quadrupedal locomotion.

1) Walking With Significant Model Uncertainty: The experiment involves the robot walking and rotating in different directions, using both adaptive and nonadaptive controllers while carrying an unknown load. To ensure a precise comparison, we establish a unit test comprising velocity and rotation commands. This test is then executed for both adaptive and nonadaptive controllers. The experiment results show that the adaptive controller provides robust locomotion, with excellent tracking error, even when carrying an unknown 5-kg load. On the other hand, the nonadaptive controller results in a considerable error in the COM height and eventually collapses under the weight of just a 3-kg load. The comparative results for the adaptive and nonadaptive controllers are shown in Fig. 6.

2) Walking on Soft Terrain: To evaluate the capability of our proposed control method in handling unknown impact models, we conducted an experiment where the robot was made for walking on double foam, which symbolizes a soft terrain. The performance of both the adaptive and nonadaptive controllers was evaluated and compared. The results are depicted in Fig. 7, representing the robot's roll angle. The figure clearly illustrates that the adaptive controller could maintain the robot's balance on the soft terrain. In contrast, the nonadaptive controller could not do so, leading to the collapse of the robot.

B. Running With Multiple Gaits

To demonstrate the superiority of our proposed approach for dynamic gaits, we conducted experiments with the robot running while carrying an unknown load. These experiments were carried out for both the trotting and bounding gaits, with an unknown load of 5 kg and 3 kg, respectively. The results of these experiments are shown in Fig. 8. It can be seen from the figure that the tracking of the COM height during the bounding gait is more unstable compared to the trotting gait, which is due to the inherent underactuated nature of the bounding gait.

C. Time-Varying Load

To demonstrate the effectiveness of our proposed adaptive force control in adapting to model uncertainty, we conducted simulations where the robot carries a time-varying load of up to 92% of its weight during walking. As shown in Fig. 9, our approach can enable the robot to adapt to time-varying uncertainty. In the simulation, the robot starts with an unknown 5-kg load. While increasing the robot's velocity, the robot is subjected to a varying external force in the z-direction that rises to 60 N, resulting in an additional unknown 11-kg load. These

[1] [Online]. Available: https://youtu.be/5t1mSh0q3lk

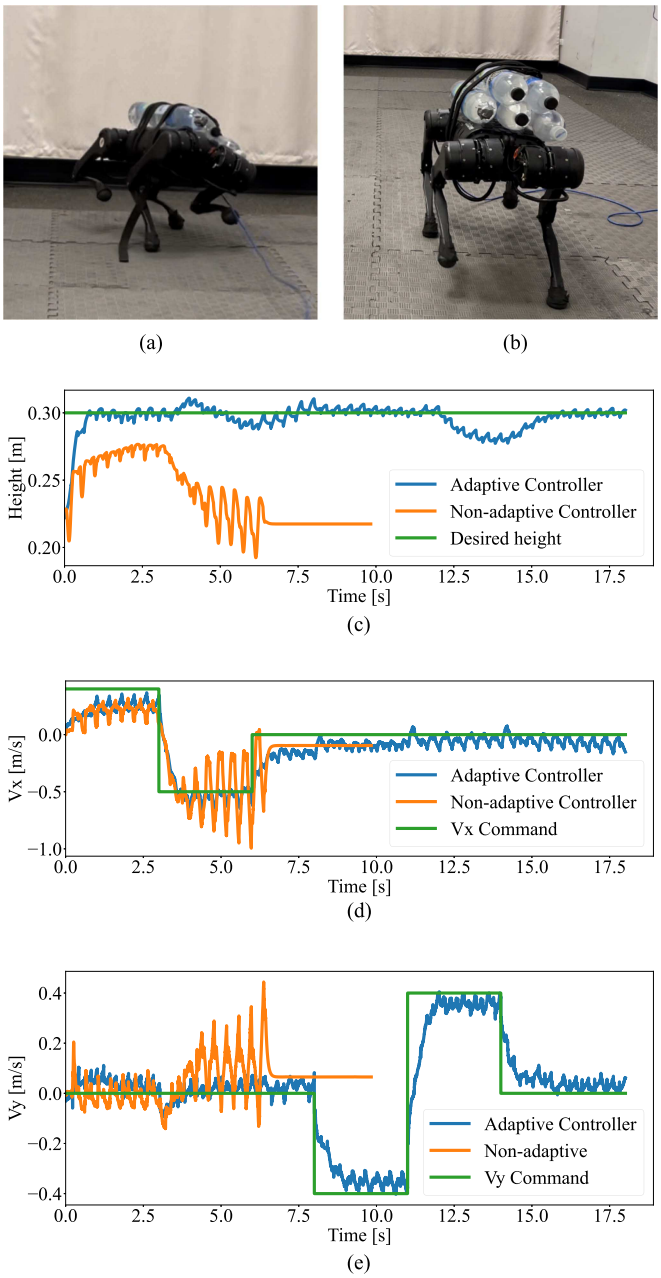


Fig. 6. *Comparing performance of adaptive and nonadaptive controllers.* (a) Snapshots of the A1 robot with the nonadaptive controller while carrying an unknown 3-kg load and collapses. (b) Snapshots of the A1 robot walking robustly with the adaptive controller while carrying an unknown 5-kg load. (c) Comparative plots of the COM height. (d) Velocity command tracking in the x-direction. (e) y-direction for adaptive and nonadaptive controllers. The plots for the nonadaptive controller do not persist until the completion of the unit test, as the robot experiences a collapse during testing.

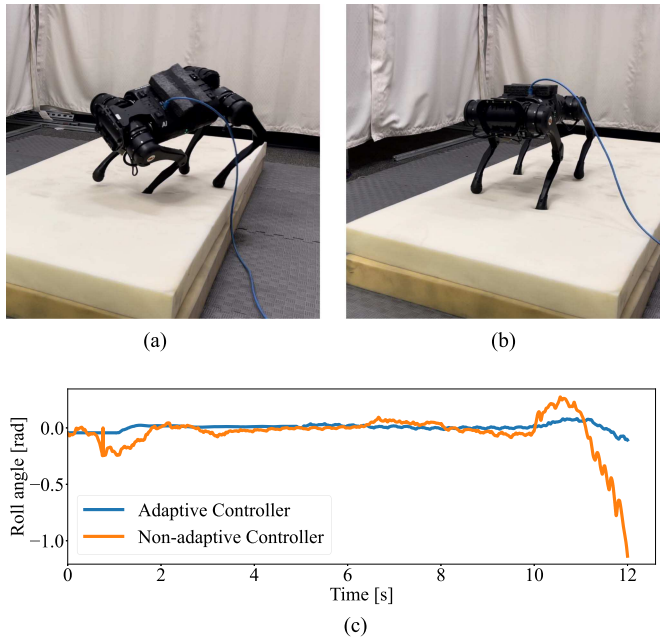


Fig. 7. *Comparing performance of adaptive and nonadaptive controllers on soft terrain.* The A1 robot tries to walk on double soft foam using (a) nonadaptive and (b) adaptive controllers. (c) Plot of the robot's roll angle.

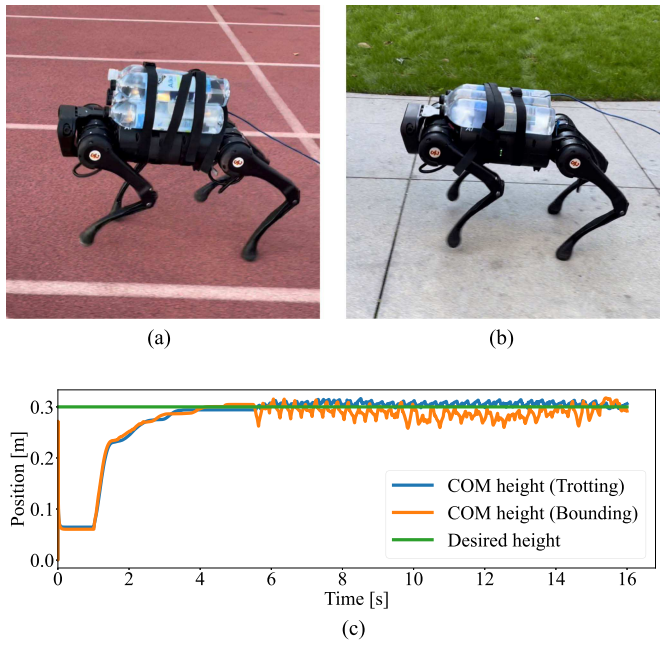


Fig. 8. *Running experiment.* The A1 robot runs with the velocity of 1 m/s using our proposed method. (a) Trotting gait with an unknown 5-kg load. (b) Bounding gait with an unknown 3-kg load. (c) Plots of COM height.

results indicate that our proposed approach effectively handles high levels of model uncertainty.

D. Terrain Uncertainty

To demonstrate the capability of our proposed method to handle terrain uncertainty, we tested the robot navigating various terrains while carrying an unknown 5-kg load. To this end, we tried walking experiments on multiple rough terrains as well as high-sloped terrain, and we got impressive results.

1) Rough Terrain: We tested the robot navigating various rough terrains such as grass and gravel. The robot walks and rotates in multiple directions while carrying an unknown 5-kg load. Some snapshots of the robot walking on diverse rough terrain are presented in Fig. 5. Our approach is based on a force controller

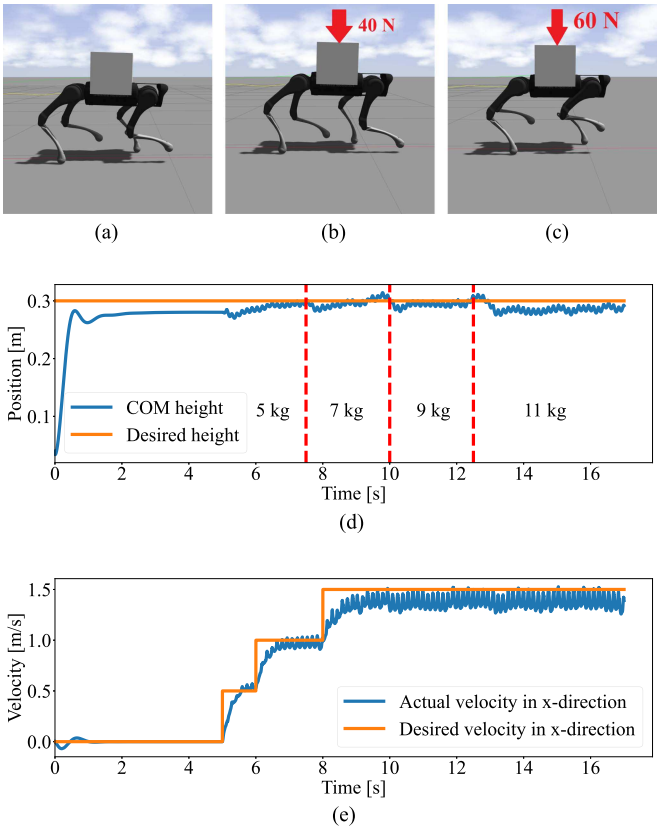


Fig. 9. *Simulation results for the robot carrying a time-varying load.* (a) Robot starts with an unknown 5-kg load, then gradually, an unknown time-varying force will be exerted on the robot as shown in (b) and (c) while the robot's velocity increases. (d) Plot of COM height. (e) Robot velocity tracking in the x-direction.

and retains the robustness features of the baseline framework, allowing the robot to handle the rough terrain effectively.

2) Sloped Terrain: To enable the robot to climb the sloped terrain perfectly without vision, we need to adjust its orientation to make its body parallel to the walking surface. This is done by using the footstep location to estimate the slope of the ground. For each ith leg, we can measure the foot position $\boldsymbol{p}_i = (p_{x,i}, p_{y,i}, p_{z,i})$ and build the vector of feet x-position ($\boldsymbol{p}_x$), y-position ($\boldsymbol{p}_y$), and z-position ($\boldsymbol{p}_z$). Thus, we can model the walking surface as a plane

$$z(x, y) = a_0 + a_1 x + a_2 y \tag{57}$$

and the coefficients (a_0, a_1, and a_2) will be obtained through the solution of the least square problem using $\boldsymbol{p}_x$, $\boldsymbol{p}_x$, and $\boldsymbol{p}_x$ data (see [56] for more details).

Note that the desired roll and pitch angles for the robot will be modified on the slope according to the following:

$$\text{roll} = \arctan(a_2), \quad \text{pitch} = \arctan(a_1). \tag{58}$$

As a result, the reference model's desired pitch and roll angles must be adjusted to the nonzero values determined as described earlier. It is important to note that the reference model utilizes the actual foot position of the robot, so there is no need to make any changes to the reference model's footstep planning when the robot is attempting to climb a slope.

VIII. Conclusion

In conclusion, a novel control system has been presented that incorporates adaptive control into force control for legged robots walking under significant uncertainties. We have demonstrated the effectiveness of our proposed approach using numerical and experimental validations. The experiments show the success of the implementation of the proposed adaptive force control on quadruped robots, allowing them to walk and run while carrying an unknown heavy load on their trunk. The results are remarkable, with the robot being able to carry a load of up to 5 kg (50% of its weight) while still keeping the tracking error within a small range and maintaining stability even in all directions. The experiment demonstrates that the proposed adaptive force control system cannot only adapt to model uncertainty but also leverage the benefits of force control in navigating rough terrains and soft terrain. On the other hand, the baseline nonadaptive controller fails to track the desired trajectory and causes the robot to collapse under uncertainty.

In the future, our goal is to broaden this methodology for autotuning MPC parameters, employing adaptive controllers specifically tailored for legged robot locomotion across various scenarios.

Appendix

A. Linear Quadratic Lyapunov Theory

According to the Lyapunov theory [71], the PD control described in (19) will asymptotically stabilize the system if

$$\boldsymbol{A}_m = \begin{bmatrix} \boldsymbol{0}_6 & \boldsymbol{1}_6 \\ -\boldsymbol{K}_P & -\boldsymbol{K}_D \end{bmatrix} \in \mathbb{R}^{12\times 12} \tag{59}$$

is Hurwitz. This means that by choosing a CLF candidate as follows:

$$V(\boldsymbol{e}) = \boldsymbol{e}^T \boldsymbol{P} \boldsymbol{e} \tag{60}$$

where $\boldsymbol{P} \in \mathbb{R}^{12\times 12}$ is the solution of the Lyapunov equation

$$\boldsymbol{A}_m{}^T \boldsymbol{P} + \boldsymbol{P} \boldsymbol{A}_m = -\boldsymbol{Q}_L \tag{61}$$

and $\boldsymbol{Q}_L \in \mathbb{R}^{12\times 12}$ is any symmetric positive-definite matrix. We then have

$$\begin{aligned} \dot{V}(\boldsymbol{e}, \boldsymbol{u}) + \lambda V(\boldsymbol{e}) = \; & \boldsymbol{e}^T \left(\boldsymbol{D}_l{}^T \boldsymbol{P} + \boldsymbol{P} \boldsymbol{D}_l \right) \boldsymbol{e} \\ & + \lambda V(\boldsymbol{e}) + 2 \boldsymbol{e}^T \boldsymbol{P} \boldsymbol{B} \boldsymbol{u} \leq 0 \end{aligned} \tag{62}$$

where,

$$\lambda = \frac{\lambda_{\min}(\boldsymbol{Q}_L)}{\lambda_{\max}(\boldsymbol{P})} > 0. \tag{63}$$

As a result, the state variable $\boldsymbol{e}$ and the control input $\boldsymbol{u}$ always remain bounded

$$\|\boldsymbol{e}\| \leq \delta_\eta, \quad \|\boldsymbol{u}\| \leq \delta_u. \tag{64}$$

However, the control signal $\boldsymbol{u}^*$ (23) we construct by solving QP problem (9) is not always the same as $\boldsymbol{u}$. Based on the friction constraints present in (9), the value of $\boldsymbol{F}^*$ is always bounded. Besides, according to the definition of $\boldsymbol{A}$, $\boldsymbol{M}$, and $\boldsymbol{G}$, these

matrices also have bounded values. Thus, it implies that

$$\|\boldsymbol{u}^*\| \leq \delta_{u^*}. \tag{65}$$

Therefore, the vector of difference between $\boldsymbol{u}$ and $\boldsymbol{u}^*$ can be defined as

$$\boldsymbol{\Delta} = \boldsymbol{u}^* - \boldsymbol{u} \tag{66}$$

which is also bounded according to (65) and (64)

$$\|\boldsymbol{\Delta}\| \leq \delta_{\Delta}. \tag{67}$$

By substituting $\boldsymbol{u}^*$ in (62), we have

$$\dot{V}(\boldsymbol{e}, \boldsymbol{u}^*) + \lambda V(\boldsymbol{e}) \leq 2\boldsymbol{e}^T \boldsymbol{P}\boldsymbol{B}\boldsymbol{\Delta} \leq \epsilon_V \tag{68}$$

where

$$\epsilon_V = 2\|\boldsymbol{P}\|\delta_{\eta}\delta_{\Delta}. \tag{69}$$

B. Stability Analysis

Theorem: Consider the system dynamics with uncertainty described by (28), and a reference model described by (29). Assume the use of an L_1 adaptive controller with the optimal closed-loop control signal given by (23), the adaptive control signal given by (33), and the adaptation laws given by (40). Then, under the aforementioned L_1 adaptive controller, the tracking error between the real model and reference model denoted as $\tilde{\boldsymbol{e}}$, as well as the errors between the real and estimated uncertainty, denoted as $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\beta}}$, respectively, are bounded.

Proof: Let us consider the following control Lyapunov candidate function:

$$\tilde{V} = \tilde{\boldsymbol{e}}^T \boldsymbol{P}\tilde{\boldsymbol{e}} + \tilde{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\alpha}} + \tilde{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\beta}}. \tag{70}$$

Therefore, its time derivative will be

$$\begin{aligned}\dot{\tilde{V}} = {} & \dot{\tilde{\boldsymbol{e}}}^T \boldsymbol{P}\tilde{\boldsymbol{e}} + \tilde{\boldsymbol{e}}^T \boldsymbol{P}\dot{\tilde{\boldsymbol{e}}} + \dot{\tilde{\boldsymbol{\alpha}}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\alpha}} + \tilde{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\alpha}}} \\ & + \dot{\tilde{\boldsymbol{\beta}}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\beta}} + \tilde{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\beta}}}\end{aligned} \tag{71}$$

in which we have

$$\begin{aligned}& \dot{\tilde{\boldsymbol{e}}}^T \boldsymbol{P}\tilde{\boldsymbol{e}} + \tilde{\boldsymbol{e}}^T \boldsymbol{P}\dot{\tilde{\boldsymbol{e}}} \\ = {} & \left(\boldsymbol{D}_l\tilde{\boldsymbol{e}} + \boldsymbol{B}\tilde{\boldsymbol{F}}\right)^T \boldsymbol{P}\tilde{\boldsymbol{e}} + \tilde{\boldsymbol{e}}^T \boldsymbol{P}\left(\boldsymbol{D}_l\tilde{\boldsymbol{e}} + \boldsymbol{B}\tilde{\boldsymbol{F}}\right) \\ & + \tilde{\boldsymbol{\alpha}}^T \boldsymbol{B}^T \|\boldsymbol{e}\| \boldsymbol{P}\tilde{\boldsymbol{e}} + \tilde{\boldsymbol{e}}^T \boldsymbol{P}\boldsymbol{B}\tilde{\boldsymbol{\alpha}}\|\boldsymbol{e}\| \\ & + \tilde{\boldsymbol{\beta}}^T \boldsymbol{B}^T \boldsymbol{P}\tilde{\boldsymbol{e}} + \tilde{\boldsymbol{e}}^T \boldsymbol{P}\boldsymbol{B}\tilde{\boldsymbol{\beta}}.\end{aligned} \tag{72}$$

Because $\tilde{\boldsymbol{e}} = \hat{\boldsymbol{e}} - \boldsymbol{e}$ satisfies the condition imposed by (68), it implies that

$$\begin{aligned}& \left(\boldsymbol{D}_l\tilde{\boldsymbol{e}} + \boldsymbol{B}\tilde{\boldsymbol{F}}\right)^T \boldsymbol{P}\tilde{\boldsymbol{e}} + \\ & \tilde{\boldsymbol{e}}^T \boldsymbol{P}\left(\boldsymbol{D}_l\tilde{\boldsymbol{e}} + \boldsymbol{B}\tilde{\boldsymbol{F}}\right) \leq -\lambda\tilde{\boldsymbol{e}}^T \boldsymbol{P}\tilde{\boldsymbol{e}} + \epsilon_{\tilde{V}}\end{aligned} \tag{73}$$

where

$$\epsilon_{\tilde{V}} = 2\|\boldsymbol{P}\|\delta_{\tilde{e}}\delta_{\tilde{\Delta}}. \tag{74}$$

Furthermore, with the property of the projection operator [63], we have the following:

$$\begin{aligned}(\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha})^T \left(\text{Proj}(\hat{\boldsymbol{\alpha}}, \boldsymbol{y}_{\alpha}) - \boldsymbol{y}_{\alpha}\right) \leq 0 \\ (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T \left(\text{Proj}(\hat{\boldsymbol{\beta}}, \boldsymbol{y}_{\beta}) - \boldsymbol{y}_{\beta}\right) \leq 0.\end{aligned} \tag{75}$$

From (40) and (75), we can imply that

$$\begin{aligned}\tilde{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\alpha}}} \leq \tilde{\boldsymbol{\alpha}}^T \boldsymbol{y}_{\alpha} - \tilde{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\dot{\boldsymbol{\alpha}} \\ \tilde{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\beta}}} \leq \tilde{\boldsymbol{\beta}}^T \boldsymbol{y}_{\beta} - \tilde{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\dot{\boldsymbol{\beta}}.\end{aligned} \tag{76}$$

We now replace (72), (73), and (76) to (71), which results in

$$\begin{aligned}\dot{\tilde{V}} \leq {} & -\lambda\tilde{\boldsymbol{e}}^T \boldsymbol{P}\tilde{\boldsymbol{e}} + \epsilon_{\tilde{V}} \\ & + \tilde{\boldsymbol{\alpha}}^T \left(\boldsymbol{y}_{\alpha} + \boldsymbol{B}^T \boldsymbol{P}\tilde{\boldsymbol{e}}\|\boldsymbol{e}\|\right) - \tilde{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\dot{\boldsymbol{\alpha}} \\ & + \left(\boldsymbol{y}_{\alpha}^T + \tilde{\boldsymbol{e}}^T \boldsymbol{P}\boldsymbol{B}\|\boldsymbol{e}\|\right)\tilde{\boldsymbol{\alpha}} - \dot{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\boldsymbol{\alpha} \\ & + \tilde{\boldsymbol{\beta}}^T \left(\boldsymbol{y}_{\beta} + \boldsymbol{B}^T \boldsymbol{P}\tilde{\boldsymbol{e}}\right) - \tilde{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\dot{\boldsymbol{\beta}} \\ & + \left(\boldsymbol{y}_{\beta}^T + \tilde{\boldsymbol{e}}^T \boldsymbol{P}\boldsymbol{B}\right)\tilde{\boldsymbol{\beta}} - \dot{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\beta}}.\end{aligned} \tag{77}$$

So, by using the chosen projection functions (41), then we conclude that

$$\begin{aligned}\dot{\tilde{V}} + \lambda\tilde{V} \leq {} & \epsilon_{\tilde{V}} + \lambda\tilde{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\alpha}} + \lambda\tilde{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\beta}} \\ & - \tilde{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\dot{\boldsymbol{\alpha}} - \dot{\boldsymbol{\alpha}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\alpha}} \\ & - \tilde{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\dot{\boldsymbol{\beta}} - \dot{\boldsymbol{\beta}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\beta}}.\end{aligned} \tag{78}$$

We assume that the uncertainties $\boldsymbol{\alpha}, \boldsymbol{\beta}$, and their time derivatives are bounded. Furthermore, the projection operators (40) will also keep $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\beta}}$ bounded (see [43] for a detailed proof about these properties). We define these bounds as follows:

$$\begin{aligned}||\tilde{\boldsymbol{\alpha}}|| \leq \tilde{\boldsymbol{\alpha}}_b, \ ||\tilde{\boldsymbol{\beta}}|| \leq \tilde{\boldsymbol{\beta}}_b \\ ||\dot{\boldsymbol{\alpha}}|| \leq \dot{\boldsymbol{\alpha}}_b, \ ||\dot{\boldsymbol{\beta}}|| \leq \dot{\boldsymbol{\beta}}_b.\end{aligned} \tag{79}$$

Combining this with (78), we have

$$\dot{\tilde{V}} + \lambda\tilde{V} \leq \lambda\delta_{\tilde{V}} \tag{80}$$

where

$$\delta_{\tilde{V}} = 2||\boldsymbol{\Gamma}||^{-1}\left(\tilde{\boldsymbol{\alpha}}_b^2 + \tilde{\boldsymbol{\beta}}_b^2 + \frac{1}{\lambda}\tilde{\boldsymbol{\alpha}}_b\dot{\boldsymbol{\alpha}}_b + \frac{1}{\lambda}\tilde{\boldsymbol{\beta}}_b\dot{\boldsymbol{\beta}}_b\right) + \frac{1}{\lambda}\epsilon_{\tilde{V}}. \tag{81}$$

Thus, if $\tilde{V} \geq \delta_{\tilde{V}}$ then $\dot{\tilde{V}} \leq 0$. As a result, we always have $\tilde{V} \leq \delta_{\tilde{V}}$. In other words, by choosing the adaptation gain $\boldsymbol{\Gamma}$ sufficiently large and $\boldsymbol{P}$ relatively small, we can limit the CLF (70) in an arbitrarily small neighborhood $\delta_{\tilde{V}}$ of the origin. According to (59) and (61), achieving a small value for $\boldsymbol{P}$ depends on choosing a proper value for $\boldsymbol{K}_P$, $\boldsymbol{K}_D$, and $\boldsymbol{Q}_L$. Therefore, the value of PD gains affects the stability of the whole system. Finally, the tracking errors between the dynamics model (28) and the reference model (29), $\tilde{\boldsymbol{e}}$, and the error between the

real and estimated uncertainty, $\tilde{\boldsymbol{\alpha}}$, $\tilde{\boldsymbol{\beta}}$ are bounded as follows:

$$||\tilde{\boldsymbol{e}}|| \leq \sqrt{\frac{\delta_{\tilde{V}}}{||\boldsymbol{P}||}}, ||\tilde{\boldsymbol{\alpha}}|| \leq \sqrt{||\boldsymbol{\Gamma}||\delta_{\tilde{V}}}, ||\tilde{\boldsymbol{\beta}}|| \leq \sqrt{||\boldsymbol{\Gamma}||\delta_{\tilde{V}}}. \quad (82)$$

ACKNOWLEDGMENT

The opinions expressed are those of the authors and do not necessarily reflect the opinions of the sponsors.

REFERENCES

[1] J. Di 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 Int. Conf. Intell. Robots Syst.*, 2018, pp. 7440–7447.

[2] M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Auton. Robots*, vol. 41, no. 1, pp. 259–272, 2017.

[3] 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. Automat.*, 2019, pp. 7448–7454.

[4] H. W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Proc. Robot.: Sci. Syst.*, vol. 11, 2015, pp. 1–9.

[5] H. W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the MIT Cheetah 2: Control design and experiments," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 167–192, 2017.

[6] H. Fukushima, T.-H. Kim, and T. Sugie, "Adaptive model predictive control for a class of constrained linear systems based on the comparison model," *Automatica*, vol. 43, no. 2, pp. 301–308, 2007.

[7] V. Adetola, D. DeHaan, and M. Guay, "Adaptive model predictive control for constrained nonlinear systems," *Syst. Control Lett.*, vol. 58, no. 5, pp. 320–326, 2009.

[8] K. Pereida and A. P. Schoellig, "Adaptive model predictive control for high-accuracy trajectory tracking in changing conditions," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, pp. 7831–7837, 2018.

[9] X. Lu, M. Cannon, and D. Koksal-Rivet, "Robust adaptive model predictive control: Performance and parameter estimation," *Int. J. Robust Nonlinear Control*, vol. 31, no. 18, pp. 8703–8724, 2021.

[10] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Proc. Conf. Robot Learn.*, vol. 100, 2020, pp. 1101–1112.

[11] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," *in Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[12] K. Y. Chee, T. Z. Jiahao, and M. A. Hsieh, "KNODE-MPC: A knowledge-based data-driven predictive control framework for aerial robots," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2819–2826, Apr., 2022.

[13] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," in *Proc. IEEE-RAS 22nd Int. Conf. Humanoid Robots (Humanoids)*, 2023, pp. 1–8.

[14] G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen, "Robust high-speed running for quadruped robots via deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 10364–10370.

[15] R. Yang, M. Zhang, N. Hansen, D. H. Xu, and X. Wang, "Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers," in *Proc. Int. Conf. Learn. Representations*, 2022.

[16] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[17] L. Bottou and Y. Le Cun, "Large scale online learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 16, 2003.

[18] A. Nagabandi, C. Finn, and S. Levine, "Deep online learning via meta-learning: Continual adaptation for model-based RL," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[19] T. Duong and N. Atanasov, "Adaptive control of SE(3) Hamiltonian dynamics with learned disturbance features," *IEEE Control Syst. Lett.*, vol. 6, pp. 2773–2778, May 2022.

[20] T. Z. Jiahao, K. Y. Chee, and M. A. Hsieh, "Online dynamics learning for predictive control with an application to aerial robots," in *Proc. 6th Conf. Robot Learn.*, vol. 205, 2023, pp. 2251–2261.

[21] S. Yang, H. Choset, and Z. Manchester, "Online kinematic calibration for legged robots," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8178–8185, Jul., 2022.

[22] S. Zhou, K. Pereida, W. Zhao, and A. P. Schoellig, "Bridging the model-reality gap with Lipschitz network adaptation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 1, pp. 642–649, Jan., 2022.

[23] A. M. Annaswamy and A. L. Fradkov, "A historical perspective of adaptive control and learning," *Annu. Rev. Control*, vol. 52, pp. 18–41, 2021.

[24] Y. Sun et al., "Online learning of unknown dynamics for model-based controllers in legged locomotion," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 8442–8449, Oct., 2021.

[25] M. Zhuang and D. Atherton, "Automatic tuning of optimum PID controllers," *Proc. Inst. Elect. Eng. D Control Theory Appl.*, vol. 140, no. 3, pp. 216–224, 1993.

[26] K. Åström, T. Hägglund, C. Hang, and W. Ho, "Automatic tuning and adaptation for PID controllers - A survey," *Control Eng. Pract.*, vol. 1, no. 4, pp. 699–714, 1993.

[27] A. R. Kumar and P. J. Ramadge, "DiffLoop: Tuning PID controllers by differentiating through the feedback loop," in *Proc. IEEE 55th Annu. Conf. Inf. Sci. Syst.*, 2021, pp. 1–6.

[28] A. Loquercio, A. Saviolo, and D. Scaramuzza, "AutoTune: Controller tuning for high-speed flight," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4432–4439, Apr., 2022.

[29] R. Calandra, N. Gopalan, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian gait optimization for bipedal locomotion," in *Learning and Intelligent Optimization*. Cham, Switzerland: Springer, 2014, pp. 274–290.

[30] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans, "Automatic gait optimization with Gaussian process regression," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 944–949.

[31] M. Mehndiratta, E. Camci, and E. Kayacan, "Can deep models help a robot to tune its controller? A step closer to self-tuning model predictive controllers," *Electronics*, vol. 10, no. 18, 2021, Art. no. 2187.

[32] S. Cheng et al., "DiffTune$^+$: Hyperparameter-free auto-tuning using auto-differentiation," in *Proc. Mach. Learn. Res.*, vol. 211, 2023, pp. 170–183.

[33] S. Cheng et al., "DiffTune: Auto-tuning through auto-differentiation," 2023, *arXiv:2209.10021*.

[34] K. J. Åström, "Adaptive control," in *Mathematical System Theory: The Influence of R. E. Kalman*, A. C. Antoulas, Ed. Berlin, Germany: Springer, 1991, pp. 437–450.

[35] J.-J. E. Slotine and W. Li, "On the adaptive control of robot manipulators," *Int. J. Robot. Res.*, vol. 6, no. 3, pp. 49–59, 1987.

[36] Y.-H. Liu and S. Arimoto, "Decentralized adaptive and nonadaptive position/force controllers for redundant manipulators in cooperations," *Int. J. Robot. Res.*, vol. 17, no. 3, pp. 232–247, 1998.

[37] Z. Li, S. S. Ge, and Z. Wang, "Robust adaptive control of coordinated multiple mobile manipulators," *Mechatronics*, vol. 18, no. 5/6, pp. 239–250, 2008.

[38] P. Culbertson, J.-J. Slotine, and M. Schwager, "Decentralized adaptive control for collaborative manipulation of rigid bodies," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1906–1920, Dec. 2021.

[39] M. Sombolestan and Q. Nguyen, "Hierarchical adaptive loco-manipulation control for quadruped robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 12156–12162.

[40] M. Sombolestan and Q. Nguyen, "Hierarchical adaptive control for collaborative manipulation of a rigid object by quadrupedal robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 2752–2759.

[41] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, vol. 199. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.

[42] D. Zhang and B. Wei, "A review on model reference adaptive control of robotic manipulators," *Annu. Rev. Control*, vol. 43, pp. 188–198, 2017.

[43] C. Cao and N. Hovakimyan, "L1 adaptive controller for a class of systems with unknown nonlinearities: Part I," in *Proc. Amer. Control Conf.*, 2008, pp. 4093–4098.

[44] C. Cao and N. Hovakimyan, "Stability margins of L1 adaptive controller: Part II," in *Proc. IEEE Amer. Control Conf.*, 2007, pp. 931–3936.

[45] C. Cao and N. Hovakimyan, "Design and analysis of a novel L1 adaptive controller, Part II: Guaranteed transient performance," in *Proc. IEEE Amer. Control Conf.*, vol. 2006, 2006, pp. 3403–3408.

[46] N. Hovakimyan and C. Cao, *L1 Adaptive Control Theory: Guaranteed Robustness With Fast Adaptation*. Philadelphia, PA, USA: SIAM, 2010, no. 5.

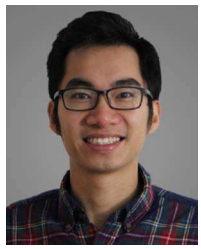
[47] A. Gahlawat, P. Zhao, A. Patterson, N. Hovakimyan, and E. Theodorou, "L1-GP: L1 adaptive control with Bayesian learning," in *Proc. 2nd Conf. Learn. Dyn. Control*, vol. 120, 2020, pp. 826–837.

[48] A. Gahlawat et al., "Contraction L1-Adaptive control using Gaussian processes," in *Proc. 3rd Conf. Learn. Dyn. Control*, vol. 144, 2021, pp. 1027–1040.

[49] G. Tournois, M. Focchi, A. Del Prete, R. Orsolino, D. G. Caldwell, and C. Semini, "Online payload identification for quadruped robots," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2017, pp. 4889–4896.
[50] Q. Nguyen and K. Sreenath, "L1 adaptive control for bipedal robots with control Lyapunov function based quadratic programs," in *Proc. Amer. Control Conf.*, vol. 2015, 2015, pp. 862–867.
[51] Q. Nguyen, A. Agrawal, W. Martin, H. Geyer, and K. Sreenath, "Dynamic bipedal locomotion over stochastic discrete terrain," *Int. J. Robot. Res.*, vol. 37, no. 13/14, pp. 1537–1553, 2018.
[52] K. Sreenath, H.-W. Park, I. Poulakakis, and J. Grizzle, "Embedding active force control within the compliant hybrid zero dynamics to achieve stable, +fast running on MABEL," *Int. J. Robot. Res.*, vol. 32, no. 3, pp. 324–345, 2013.
[53] J. W. Grizzle, C. Chevallereau, and C. L. Shih, "HZD-based control of a five-link underactuated 3D bipedal robot," in *Proc. IEEE Conf. Decis. Control*, 2008, pp. 5206–5213.
[54] M. V. Minniti, R. Grandia, F. Farshidian, and M. Hutter, "Adaptive CLF-MPC with application to quadrupedal robots," *IEEE Robot. Autom. Lett.*, vol. 7, no. 1, pp. 565–572, Jan. 2022.
[55] M. Sombolestan, Y. Chen, and Q. Nguyen, "Adaptive force-based control for legged robots," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2021, pp. 7440–7447.
[56] G. Bledt et al., "MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2018, pp. 2245–2252.
[57] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of dynamic gaits for a quadrupedal robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 3287–3292.
[58] F. Bullo and R. M. Murray, "Proportional derivative (PD) control on the Euclidean group," 1995. [Online]. Available: https://resolver.caltech.edu/CaltechCDSTR:1995.CIT-CDS-95-010
[59] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC, 2017.
[60] M. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT Press, 1986.
[61] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Proc. IEEE-RAS 36th Int. Conf. Humanoid Robots*, 2006, pp. 200–207.
[62] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Trans. Autom. Control*, vol. 48, no. 1, pp. 42–56, Jan. 2003.
[63] E. Lavretsky and T. E. Gibson, "Projection operator in adaptive systems," 2011, *arXiv:1112.4232*.
[64] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.
[65] M. S. Fadali and A. Visioli, *Digital Control Engineering: Analysis and Design*. Amsterdam, The Netherlands: Elsevier Science, 2012.
[66] "Boost odeint library." [Online]. Available: https://www.boost.org/doc/libs/1_64_0/libs/numeric/odeint/doc/html/index.html
[67] M. Azad and M. N. Mistry, "Balance control strategy for legged robots with compliant contacts," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4391–4396.
[68] V. Vasilopoulos, I. S. Paraskevas, and E. G. Papadopoulos, "Monopod hopping on compliant terrains," *Robot. Auton. Syst.*, vol. 102, pp. 13–26, 2018.
[69] A. H. Chang, C. M. Hubicki, J. J. Aguilar, D. I. Goldman, A. D. Ames, and P. A. Vela, "Learning to jump in granular media: Unifying optimal control synthesis with Gaussian process-based regression," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 2154–2160.
[70] S. Fahmi, M. Focchi, A. Radulescu, G. Fink, V. Barasuol, and C. Semini, "STANCE: Locomotion adaptation over soft terrain," *IEEE Trans. Robot.*, vol. 36, no. 2, pp. 443–457, Apr. 2020.
[71] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 876–891, Apr. 2014.

Mohsen Sombolestan received the B.Sc. degree in mechanical engineering from the Sharif University of Technology, Tehran, Iran, in 2017, and the M.Sc. degree in mechanical engineering from the Isfahan University of Technology, Isfahan, Iran, in 2020. He is currently working toward the Ph.D. degree in mechanical engineering from the University of Southern California, Los Angeles, CA, USA.

His research interests include control system design in robotic applications, especially legged robots, focusing on adaptive control, model predictive control, and reinforcement learning.



Quan Nguyen received the Ph.D. degree in mechanical engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA, in 2017, with the Best Dissertation Award.

He is currently an Assistant Professor of Aerospace and Mechanical Engineering with the University of Southern California (USC), Los Angeles, CA, USA. Before joining USC, he was a Postdoctoral Associate with Biomimetic Robotics Lab, Massachusetts Institute of Technology (MIT). His research interests span different control and optimization approaches for highly dynamic robotics, including nonlinear control, trajectory optimization, real-time optimization-based control, and robust and adaptive control.

Dr. Nguyen won the Best Presentation of the Session at the 2016 American Control Conference and the Best System Paper Finalist at the 2017 Robotics: Science & Systems Conference. His work on the MIT Cheetah 3 robot leaping on a desk was featured widely in many major media channels, including CNN, BBC, NBC, ABC, etc.