IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Adaptive Legged Locomotion via Online Learning for Model Predictive Control

Hongyu Zhou, Xiaoyu Zhang, and Vasileios Tzoumas, *Senior Member, IEEE*

***Abstract*—We provide an algorithm for adaptive legged locomotion via online learning and model predictive control. The algorithm is composed of two interacting modules: model predictive control (MPC) and online learning of residual dynamics. The residual dynamics can represent modeling errors and external disturbances. We are motivated by the future of autonomy where quadrupeds will autonomously perform complex tasks despite real-world unknown uncertainty, such as unknown payload and uneven terrains. The algorithm uses random Fourier features to approximate the residual dynamics in reproducing kernel Hilbert spaces. Then, it employs MPC based on the current learned model of the residual dynamics. The model is updated online in a self-supervised manner using least squares based on the data collected while controlling the quadruped. The algorithm enjoys sublinear *dynamic regret*, defined as the suboptimality against an optimal clairvoyant controller that knows how the residual dynamics. We validate our algorithm in Gazebo and MuJoCo simulations, where the quadruped aims to track reference trajectories. The Gazebo simulations include constant unknown external forces up to $12g$, where g is the gravity vector, in flat terrain, slope terrain with 20° inclination, and rough terrain with 0.25 m height variation. The MuJoCo simulations include time-varying unknown disturbances with payload up to 8 kg and time-varying ground friction coefficients in flat terrain.**

***Index Terms*—Legged control, online learning, adaptive model predictive control, random feature approximation.**

I. Introduction

LEGGED robots promise to automate essential tasks such as search and rescue, payload delivery, and industrial inspection [1]. Successfully accomplishing these tasks necessitates accurate and efficient tracking. However, achieving both accuracy and efficiency in legged locomotion is challenging due to uncertainties arising from the robot's imperfect model and environmental disturbances. For example, quadrupeds need to (i) pick up and transport packages of unknown weight and (ii) navigate diverse terrains with varying friction and elevation.

Received 15 June 2025; accepted 25 November 2025. Date of publication 15 December 2025; date of current version 23 December 2025. This article was recommended for publication by Associate Editor C. Mastalli and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by NSF CAREER under Grant 2337412 and in part by Rackham Predoctoral Fellowship of the University of Michigan. *(Hongyu Zhou and Xiaoyu Zhang contributed equally to this work.) (Corresponding author: Hongyu Zhou.)*

Hongyu Zhou and Vasileios Tzoumas are with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: zhouhy@umich.edu; vtzoumas@umich.edu).

Xiaoyu Zhang is with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: xzhang636@gatech.edu).

The code is open-sourced at https://github.com/UM-iRaL/Adaptive-Legged-Locomotion.

Digital Object Identifier 10.1109/LRA.2025.3644161

State-of-the-art methods for legged control under uncertainties (*i.e.*, residual dynamics) typically rely on reinforcement learning [2], [3], [4], [5], [6], [7], [8], [9], [10] or robust and adaptive control [11], [12], [13], [14], [15], [16]. The reinforcement learning methods require offline training and high-fidelity simulators, which can be costly and time-consuming. The robust control methods can be conservative due to the assumption of worst-case disturbance realization and can be computationally expensive for real-time control of legged locomotion. The adaptive control methods assume parametric uncertainty additive to the known system dynamics and update these coefficients online to enhance the robustness against disturbances, but they often assume the uncertainty to be either vector-valued or a linear function of the state.

In this letter, instead, we leverage the success of online learning and model predictive control methods for accurate tracking control under uncertainty [17]. To this end, we learn online a predictive model of the residual dynamics in a self-supervised manner (Fig. 2). Therefore, the method achieves: one-shot online learning, instead of offline; online control that adapts to the actual disturbance realization, instead of the worst-case; and model predictive control (MPC) with a learned residual dynamics model in a reproducing kernel Hilbert space (RKHS), instead of vector-valued or linear functions. We elaborate on our contributions below.

Contributions: We provide a real-time and asymptotically-optimal algorithm for adaptive legged locomotion under unknown residual dynamics. The algorithm is composed of two interacting modules (Fig. 2): (i) a MPC module, and (ii) an online learning module. At each time step, the MPC module uses the learned residual dynamics model from the online learning module to calculate the next control input. Given the control input and the observed new state, the online learning module then updates the residual dynamics model. The update in the online learning module is based on online least-squares estimation via online gradient descent (OGD) [18], where the residual dynamics in RKHS [19] are parameterized as a linear combination of random Fourier features [20], [21], [22]. This allows us to learn residual dynamics from a rich functional class while maintaining the computational efficiency of classical finite-dimensional parametric approximations that can be used in MPC. The algorithm is asymptotically-optimal in the sense of achieving sublinear *dynamic regret*, defined as the suboptimality against an optimal clairvoyant controller that knows how the unknown residual dynamics.

Numerical Experiments: We validate the algorithm in simulated scenarios in Gazebo with uncertainty up to $12\boldsymbol{g}$, where $\boldsymbol{g}$ is the gravity vector. We test the algorithm in flat terrain, slope terrain with 20° inclination, and rough terrain with 0.25 m height variation. We compare our algorithm with a nominal

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

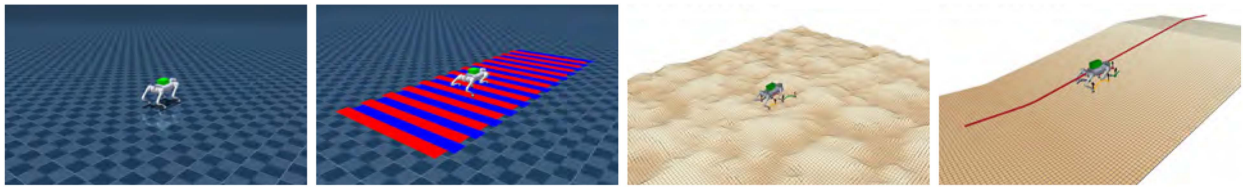


Fig. 1. Adaptive Legged Locomotion via Online Learning and Model Predictive Control: Tested Scenarios of Reference Trajectory Tracking subject to Various Unknown Disturbances. In this letter, we focus on adaptive legged locomotion via online learning and model predictive control, where the quadruped aims to achieve accurate reference tracking control despite unknown dynamics and external disturbances (*i.e.*, residual dynamics). For example, the reference tracking performance of the quadruped can be challenged by *various terrain* (first: flat terrain with fixed friction coefficient & second: flat terrain with changing friction coefficients (indicate by colors) & third: rough terrain & fourth: slope terrain with 20° inclination) and *payloads with unknown weights* (all). We provide a control algorithm that demonstrates adaptive legged locomotion by learning such unknown dynamics/disturbances online based on the data collected at runtime, and using the learned model of residual dynamics for predictive control.

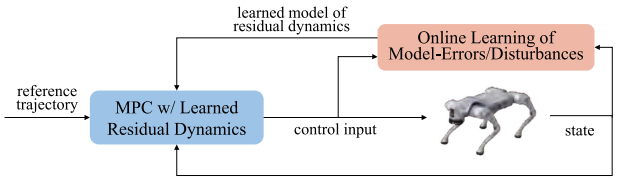


Fig. 2. Architecture of Adaptive Legged Locomotion via Online Learning and Model Predictive Control. The pipeline has two modules: (i) a model predictive control (MPC) module, and (ii) an online learning module. The MPC module uses the learned residual dynamics model from the online learning module to calculate the next control input. Given the control input and the observed new state, the online learning module updates the residual dynamics model.

MPC that ignores the unknown dynamics or disturbances, and a heuristic L1 MPC (L1-MPC) that uses an estimated vector value of uncertainty across the MPC horizon. The algorithm (i) achieves up to 67% improvement of tracking performance over the nominal MPC and 21% improvement over L1-MPC, and (ii) succeeds even when the nominal MPC fails, despite the uncertainty due to external forces and challenging terrains. We validate the algorithm in simulated scenarios in MuJoCo under time-varying uncertainty in flat terrains of different ground coefficients with up to 8 kg payload, showing the algorithm achieves significantly better tracking performance than Nominal MPC.

II. Related Work

We discuss related work on legged control using (i) reinforcement learning and (ii) robust and adaptive control, and related work on online learning for control.

Reinforcement learning: Reinforcement learning algorithms have achieved dynamic legged locomotion by training control policy in a simulator with massive parallelization [2], [3], [4], [5], [6], [7], [8], [9], [10]. Despite their success, they often face sim-to-real gap when deploying the policy trained in simulator to hardware. To address this, a wide range of environment parameters and sensor noises are used to learn the control policy which is robust in this range [2], [4], [5], [6]. However, domain randomization trades optimality for robustness, leading to a conservative control policy [23]. Alternatively, a high-fidelity simulator built by using real-world robot data can be used [2], [3], but it can be time-consuming and not transferable to different robots. [7], [8], [9], [10] train an environment encoder to obtain environmental information based on onboard sensors in a latent space that is used for the trained policy for adaptation. In this letter, instead, our algorithm requires no offline data collection and training, achieving one-shot online learning in a self-supervised manner based on data collected online and adapting to real-world environments on-the-spot.

Robust and adaptive control: Robust control methods [24], [25] select control inputs assuming the worst-case realization of disturbances. However, assuming the worst-case disturbances can be conservative. In addition, these approaches are also computationally expensive, thus limiting their applications to real-time control of legged locomotion [11], [12], where convexification of the MPC problem or a specialized MPC solver is required. Adaptive control methods [26], [27], [28] often assume parametric uncertainty additive to the known system dynamics and update these coefficients online to enhance the robustness against disturbances. Our method falls into the class of adaptive control methods and learns a model of disturbances online to be used in MPC. Notably, our method (i) requires no offline system identification of basis functions as required in [13], and (ii) learns a function in a Reproducing Kernel Hilbert Space (RKHS) using random Fourier features, in contrast to a linear function in [14], [15], [16].

Online learning for control: Online learning algorithms based on online convex optimization (OCO) [17], [18], [29], [30], [31], [32], [33], [34], [35] consider the control problem as a sequential game between a controller and an environment. They quantify the control performance through *regret*, *i.e.*, the sub-optimality against an optimal clairvoyant controller that knows the unknown disturbances and dynamics. [29], [30], [31], [32] update control inputs based on observed disturbances only since they assume no model that can be used to simulate the future evolution of the disturbances. The proposed approaches have been observed to be sensitive to the tuning parameters in [30]. Instead of optimizing the online controller, [17], [34], [35] learn the model of the disturbances, and use model predictive control to select control inputs based on the learned disturbance model.

III. Adaptive Legged Locomotion via Online Learning and Model Predictive Control

We formulate the problem of *Adaptive Legged Locomotion via Online Learning and Model Predictive Control* (Problem 1). We use the following framework and assumptions.

Single Rigid Body Dynamics: We consider the single rigid body dynamics of the form

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \quad m\dot{\boldsymbol{v}} = m\boldsymbol{g} + \boldsymbol{R}\sum_{i=1}^{4} \boldsymbol{f}_i + \boldsymbol{f}_u,$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

$$\dot{\boldsymbol{\theta}} = \boldsymbol{T}(\boldsymbol{\theta})\,\boldsymbol{\omega}, \quad \boldsymbol{\mathcal{J}}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \boldsymbol{\mathcal{J}}\boldsymbol{\omega} + \sum_{i=1}^{4} \boldsymbol{r}_i \times \boldsymbol{f}_i + \boldsymbol{\tau}_u, \tag{1}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ and $\boldsymbol{v} \in \mathbb{R}^3$ are position and velocity in the inertial frame, $\boldsymbol{\theta}$ is the Euler angle, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the body angular velocity, m is the mass, $\boldsymbol{\mathcal{J}}$ is the inertia matrix, $\boldsymbol{g}$ is the gravity vector, $\boldsymbol{R} \in SO(3)$ is the rotation matrix from the body to inertial frame, $\boldsymbol{T} : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ is the Euler angle transformation matrix, $\boldsymbol{f}_i \in \mathbb{R}^3$ is the contact force of the i-th foot, $\boldsymbol{r}_i \in \mathbb{R}^3$ is the i-th foot's position in body frame, $\boldsymbol{f}_u \in \mathbb{R}^3$ is the unknown force in inertial frame, and $\boldsymbol{\tau}_u \in \mathbb{R}^3$ is the unknown torque in body frame.

For convenience, we rewrite (1) into the form of a control-affine system as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ \boldsymbol{J}^\top \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ \boldsymbol{h}(\boldsymbol{z}) \end{bmatrix}, \tag{2}$$

where $\boldsymbol{x} \triangleq [\boldsymbol{p}^\top\ \boldsymbol{\theta}^\top\ \boldsymbol{v}^\top\ \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^{12}$ is the state, $\boldsymbol{u} \triangleq [\boldsymbol{f}_1^\top\ \boldsymbol{f}_2^\top\ \boldsymbol{f}_3^\top\ \boldsymbol{f}_4^\top]^\top \in \mathbb{R}^{12}$ is the control input, $\boldsymbol{f} : \mathbb{R}^{12} \to \mathbb{R}^{12}$ is a known locally Lipschitz function, $\boldsymbol{J} \in \mathbb{R}^{12\times6}$ is the contact Jacobian matrix that depends on $\boldsymbol{x}$ and $\boldsymbol{r}_i$, $\boldsymbol{h} \triangleq [\boldsymbol{f}_u^\top\ \boldsymbol{\tau}_u^\top]^\top : \mathbb{R}^{d_z} \to \mathbb{R}^6$ is the unknown disturbances, and $\boldsymbol{z} \in \mathbb{R}^{d_z}$ is a vector of features chosen as a subset of $[\boldsymbol{x}^\top\ \boldsymbol{u}^\top]^\top$. $\boldsymbol{h}(\cdot)$ represents unknown residual dynamics that depend on system state and control input, which can be used to model the effects of unknown payload and uneven terrains.

Using forward Euler discretization, we can obtain the discrete-time system dynamics from (2):

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t) + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ \boldsymbol{J}_t^\top \end{bmatrix} \boldsymbol{u}_t + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ \boldsymbol{h}(\boldsymbol{z}_t) \end{bmatrix}, \tag{3}$$

where we overload the notations used in (2) and omit the discretization time. We refer to the undisturbed system dynamics as the *nominal dynamics*.

Model Predictive Control (MPC): MPC selects a control input $\boldsymbol{u}_t$ by simulating the system dynamics over a look-ahead horizon N. In the presence of unknown residual dynamics, MPC can utilize an estimate of $\boldsymbol{h}(\cdot)$:

$$\min_{\boldsymbol{x}_{t+1:t+N},\, \boldsymbol{u}_{t:t+N-1}} \sum_{k=t}^{t+N-1} c_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4a}$$

$$\textit{subject to} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ \boldsymbol{J}_k^\top \end{bmatrix} \boldsymbol{u}_k + \begin{bmatrix} \boldsymbol{0}_{6\times1} \\ \hat{\boldsymbol{h}}(\boldsymbol{z}_k) \end{bmatrix}, \tag{4b}$$

$$\boldsymbol{u}_k \in \mathcal{U}, \quad k \in \{t, \ldots, t+N-1\}, \tag{4c}$$

where $c_t(\cdot,\cdot) : \mathbb{R}^{12} \times \mathbb{R}^{12} \to \mathbb{R}$ is the cost function, $\mathcal{U}$ is a compact set that represents constraints on the control input due to, *e.g.*, controller saturation and friction cone, $\hat{\boldsymbol{h}}(\cdot)$ is the estimate of $\boldsymbol{h}(\cdot)$. Specifically, $\hat{\boldsymbol{h}}(\cdot) \triangleq \hat{\boldsymbol{h}}(\cdot;\hat{\boldsymbol{\alpha}})$ where $\hat{\boldsymbol{\alpha}}$ is a parameter that is updated online by our proposed method to improve the control performance.

Control Performance Metric: We design $\boldsymbol{u}_t$ to ensure a control performance that is comparable to an optimal clairvoyant (non-causal) policy that knows the disturbance function $\boldsymbol{h}$ a priori. Particularly, we consider the metric below.

Definition 1 (Dynamic Regret): Assume a total time horizon of operation T, and loss functions c_t, $t = 1, \ldots, T$. Then, *dynamic regret* is defined as

$$\mathrm{Regret}_T^D = \sum_{t=1}^{T} c_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{h}(\boldsymbol{z}_t)) - \sum_{t=1}^{T} c_t(\boldsymbol{x}_t^\star, \boldsymbol{u}_t^\star, \boldsymbol{h}(\boldsymbol{z}_t^\star)), \tag{5}$$

where we made the dependence of the cost c_t to the unknown disturbance $\boldsymbol{h}$ explicit, $\boldsymbol{u}_t^\star$ is the optimal control input in hindsight, *i.e.*, the optimal (non-causal) input given a priori knowledge of the unknown function $\boldsymbol{h}$ and $\boldsymbol{x}_{t+1}^\star$ is the state reached by applying the optimal control inputs $(\boldsymbol{u}_1^\star, \ldots, \boldsymbol{u}_t^\star)$.

Problem 1 (Adaptive Legged Locomotion via Online Learning and Model Predictive Control): At each $t = 1, \ldots, T$, estimate the unknown dynamics $\hat{\boldsymbol{h}}(\cdot)$, and identify a control input $\boldsymbol{u}_t$ by solving (4), such that Regret_T^D is sublinear.

A sublinear dynamics regret means $\lim_{T\to\infty} \mathrm{Regret}_T^D / T \to 0$, which implies the algorithm asymptotically converges to the optimal (non-causal) controller.

IV. Algorithm and Regret Guarantee

We present the algorithm for Problem 1 (Algorithm 1) and its performance guarantee. The algorithm is sketched in Fig. 2. The algorithm is composed of two interacting modules: (i) an MPC module, and (ii) an online system identification module. At each $t = 1, 2, \ldots$, the MPC module uses the estimated $\hat{\boldsymbol{h}}(\cdot)$ from the system identification module to calculate the control input $\boldsymbol{u}_t$. Given the current control input $\boldsymbol{u}_t$ and the observed new state $\boldsymbol{x}_{t+1}$, the online system identification module updates the estimate $\hat{\boldsymbol{h}}(\cdot)$. To this end, it employs online least-squares estimation via online gradient descent, where $\boldsymbol{h}(\cdot)$ is parameterized as a linear combination of random fourier features. To rigorously present the algorithm, we thus first introduce random Fourier features for approximating an $\boldsymbol{h}(\cdot)$ (Section IV-A), and online gradient descent for estimation (Section IV-B).

A. Function Approximation via Random Fourier Features

We overview the randomized approximation algorithm in [22] for approximating an $\boldsymbol{h}(\cdot)$. The algorithm is based on random Fourier features [20], [21] and their extension to vector-valued functions [36], [37]. By being randomized, the algorithm is computationally efficient while retaining the expressiveness of the RKHS with high probability.

Based on the assumptions of $\boldsymbol{h} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ lies in a subspace of a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ [38] and the Operator-Valued Bochner's Theorem [36], we assume that $\boldsymbol{h}$ can be written as $\boldsymbol{h}(\cdot) = \int_\Theta \boldsymbol{\Phi}(\cdot,\boldsymbol{\theta})\boldsymbol{\alpha}(\boldsymbol{\theta})\mathrm{d}\nu(\boldsymbol{\theta})$, [38] and there exists a finite-dimensional approximation of $\boldsymbol{h}(\cdot)$ by $\boldsymbol{h}(\cdot) \approx \hat{\boldsymbol{h}}(\cdot;\boldsymbol{\alpha}) \triangleq \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\Phi}(\cdot,\boldsymbol{\theta}_i)\boldsymbol{\alpha}_i$, where $\boldsymbol{\Phi}(\boldsymbol{z},\boldsymbol{\theta}) = \boldsymbol{B}(\boldsymbol{w})\phi(\boldsymbol{w}^\top\boldsymbol{z} + b)$ is the feature map, $\boldsymbol{B} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x \times d_1}$, $\phi : \mathbb{R} \to [-1, 1]$ is a 1-Lipschitz function, $d_1 \le d_x$, $\boldsymbol{\theta}_i \sim \nu$ are drawn i.i.d. from the base measure ν with $\boldsymbol{\theta} = (\boldsymbol{w}, b)$, $\boldsymbol{w} \in \mathbb{R}^{d_z}$, and $b \in \mathbb{R}$, $\boldsymbol{\alpha}_i \triangleq \boldsymbol{\alpha}(\theta_i)$ are parameters to be learned, and M is the number of sampling points that decides the approximation accuracy.

The following shows the expressiveness of the finite-dimensional approximation of $\boldsymbol{h}(\cdot)$, considering $\boldsymbol{\alpha}_i \in \mathcal{D}$, where $\mathcal{D} \triangleq \{\boldsymbol{\alpha} \mid \|\boldsymbol{\alpha}\| \le B_h\}$.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Proposition 1 (Uniformly Approximation Error [22]): Assume $\boldsymbol{h} \in \mathcal{F}_2(B_h)$, where

$$\mathcal{F}_2\left(B_h\right) \triangleq \left\{ \boldsymbol{h}\left(\cdot\right) = \int_{\Theta} \boldsymbol{\Phi}\left(\cdot, \boldsymbol{\theta}\right) \boldsymbol{\alpha}(\boldsymbol{\theta}) \mathrm{d}\nu(\boldsymbol{\theta}) \middle| \boldsymbol{\alpha} \in \mathcal{D} \right\}.$$

Let $\delta \in (0,1)$. With probability at least $1-\delta$, there exist $\{\boldsymbol{\alpha}_i\}_{i=1}^M \in \mathcal{D}$, *i.e.*, $\|\boldsymbol{\alpha}_i\| \leq B_h$, such that

$$\left\| \boldsymbol{h}\left(\cdot\right) - \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{\Phi}\left(\cdot, \boldsymbol{\theta}_i\right) \boldsymbol{\alpha}_i \right\|_{\infty} \leq \mathcal{O}\left(\frac{1}{\sqrt{M}}\right). \tag{6}$$

Proposition 1, therefore, indicates that the uniformly approximation error scales $\mathcal{O}(\frac{1}{\sqrt{M}})$.

Random Fourier features can be viewed as linearizations of neural networks [39], [40]. Neural networks, in principle, can perform better than kernel methods due to greater expressivity. However, using neural networks poses challenges in such online learning settings due to their data-hungry nature. In addition, using neural networks in MPC can be computationally expensive for embedded systems, and a customized solver or dynamics representation is required [41], [42]. Therefore, we utilize random Fourier features to balance computational efficiency and expressiveness.

B. Online Least-Squares Estimation

Given a data point $(\boldsymbol{z}_t,\ \boldsymbol{h}(\boldsymbol{z}_t))$ observed at time t, we employ an online least-squares algorithm that updates the parameters $\hat{\boldsymbol{\alpha}}_t \triangleq [\boldsymbol{\alpha}_{i,t}^\top,\ \ldots,\ \boldsymbol{\alpha}_{M,t}^\top]^\top$ to minimize the approximation error $l_t = \|\boldsymbol{h}(\boldsymbol{z}_t) - \hat{\boldsymbol{h}}(\boldsymbol{z}_t)\|^2$, where $\hat{\boldsymbol{h}}(\cdot) \triangleq \frac{1}{M}\sum_{i=1}^M \boldsymbol{\Phi}(\cdot,\boldsymbol{\theta}_i)\hat{\boldsymbol{\alpha}}_{i,t}$ and $\boldsymbol{\Phi}(\cdot,\boldsymbol{\theta}_i)$ is the random Fourier feature as in Section IV-A. Specifically, the algorithm used the online gradient descent algorithm (OGD) [18]. At each $t = 1, \ldots, T$, it makes the steps:

- Given $(\boldsymbol{z}_t,\ \boldsymbol{h}(\boldsymbol{z}_t))$, formulate the estimation loss function (approximation error): $l_t(\hat{\boldsymbol{\alpha}}_t) \triangleq \|\boldsymbol{h}(\boldsymbol{z}_t) - \frac{1}{M}\sum_{i=1}^M \boldsymbol{\Phi}(\boldsymbol{z}_t,\boldsymbol{\theta}_i)\hat{\boldsymbol{\alpha}}_{i,t}\|^2$.
- Calculate the gradient of $l_t(\hat{\boldsymbol{\alpha}}_t)$ with respect to $\hat{\boldsymbol{\alpha}}_t$: $\nabla_t \triangleq \nabla_{\hat{\boldsymbol{\alpha}}_t} l_t(\hat{\boldsymbol{\alpha}}_t)$.
- Update using gradient descent with learning rate η: $\hat{\boldsymbol{\alpha}}'_{t+1} = \hat{\boldsymbol{\alpha}}_t - \eta\nabla_t$.
- Project each $\hat{\boldsymbol{\alpha}}'_{i,t+1}$ onto $\mathcal{D}$: $\hat{\boldsymbol{\alpha}}_{i,t+1} = \Pi_{\mathcal{D}}(\hat{\boldsymbol{\alpha}}'_{i,t+1}) \triangleq \underset{\boldsymbol{\alpha}\in\mathcal{D}}{argmin}\ \|\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}'_{i,t+1}\|^2$.

The above online least-squares estimation algorithm enjoys an $\mathcal{O}(\sqrt{T})$ regret bound, per the regret bound of OGD [18].

Proposition 2 (Regret Bound of Online Least-Squares Estimation [18]): Assume $\eta = \mathcal{O}(1/\sqrt{T})$. Then,

$$\mathrm{Regret}_T^S \triangleq \sum_{t=1}^{T} l_t\left(\boldsymbol{\alpha}_t\right) - \sum_{t=1}^{T} l_t\left(\boldsymbol{\alpha}^\star\right) \leq \mathcal{O}\left(\sqrt{T}\right), \tag{7}$$

where $\boldsymbol{\alpha}^\star \triangleq \underset{}{argmin}\ \sum_{t=1}^T l_t(\boldsymbol{\alpha})$ is the optimal parameter that achieves lowest cumulative loss in hindsight.

The online least-squares estimation algorithm thus asymptotically achieves the same estimation error as the optimal parameter $\boldsymbol{\alpha}^\star$ since $\lim_{T\to\infty} \mathrm{Regret}_T^S / T = 0$.

Algorithm 1: Adaptive Legged Locomotion via Online Learning and Model Predictive Control.

Input: Number of random Fourier features M; base measure ν; domain set $\mathcal{D}$; gradient descent learning rate η.
Output: Ground reaction forces $\boldsymbol{u}_t$.
1: Initialize $\boldsymbol{x}_1$, $\hat{\boldsymbol{\alpha}}_{i,1} \in \mathcal{D}$;
2: Randomly sample $\boldsymbol{\theta}_i \sim \nu$ and formulate $\boldsymbol{\Phi}(\cdot,\boldsymbol{\theta}_i)$, where $i \in \{1,\ldots,M\}$;
3: **for** each time step $t = 1, \ldots, T$ **do**
4: Receive contact schedule, desired foothold positions, and reference trajectory;
5: Formulate (4) with $\hat{\boldsymbol{h}}(\cdot) \triangleq \frac{1}{M}\sum_{i=1}^M \boldsymbol{\Phi}(\cdot,\boldsymbol{\theta}_i)\hat{\boldsymbol{\alpha}}_{i,t}$;
6: Obtain ground reaction forces $\boldsymbol{u}_t$ by solving (4) and send $\boldsymbol{u}_t$ to low-level leg controller;
7: Observe state $\boldsymbol{x}_{t+1}$, and calculate $\boldsymbol{h}(\boldsymbol{z}_t)$ via (3);
8: Formulate estimation loss $l_t(\hat{\boldsymbol{\alpha}}_t) \triangleq \|\boldsymbol{h}(\boldsymbol{z}_t) - \frac{1}{M}\sum_{i=1}^M \boldsymbol{\Phi}(\boldsymbol{z}_t,\boldsymbol{\theta}_i)\hat{\boldsymbol{\alpha}}_{i,t}\|^2$;
9: Calculate gradient $\nabla_t \triangleq \nabla_{\hat{\boldsymbol{\alpha}}_t} l_t(\hat{\boldsymbol{\alpha}}_t)$;
10: Update $\hat{\boldsymbol{\alpha}}'_{t+1} = \hat{\boldsymbol{\alpha}}_t - \eta\nabla_t$;
11: Project $\hat{\boldsymbol{\alpha}}'_{i,t+1}$ onto $\mathcal{D}$, *i.e.*, $\hat{\boldsymbol{\alpha}}_{i,t+1} = \Pi_{\mathcal{D}}(\hat{\boldsymbol{\alpha}}'_{i,t+1})$, for $i \in \{1,\ \ldots,\ M\}$;
12: **end for**

C. Algorithm for Problem 1

The pseudo-code is given in Algorithm 1. The algorithm is composed of three steps, initialization, control, and online learning, where the control and online learning steps influence each other at each time steps (Fig. 2):

Initialization steps: Algorithm 1 first initializes the system state $\boldsymbol{x}_1$ and parameter $\hat{\boldsymbol{\alpha}}_1 \in \mathcal{D}$ (line 1). Then given the number of random Fourier features, Algorithm 1 randomly samples θ_i and formulates $\boldsymbol{\Phi}(\cdot,\boldsymbol{\theta}_i)$, where $i \in \{1,\ldots,M\}$ (line 2).

Control steps: Then, at each t, given the current estimate $\hat{\boldsymbol{h}}(\cdot) \triangleq \frac{1}{M}\sum_{i=1}^M \boldsymbol{\Phi}(\cdot,\boldsymbol{\theta}_i)\hat{\boldsymbol{\alpha}}_{i,t}$, contact schedule, desired foothold positions, and reference trajectory, Algorithm 1 applies the ground reaction forces $\boldsymbol{u}_t$ obtained by solving (4) (lines 4-6).

Learning steps: The system then evolves to state $\boldsymbol{x}_{t+1}$, and, $\boldsymbol{h}(\boldsymbol{z}_t)$ is calculated upon observing $\boldsymbol{x}_{t+1}$ (line 7). Afterwards, the algorithm formulates the loss $l_t(\hat{\boldsymbol{\alpha}}_t) \triangleq \|\boldsymbol{h}(\boldsymbol{z}_t) - \sum_{i=1}^M \boldsymbol{\Phi}(\boldsymbol{z}_t,\boldsymbol{\theta}_i)\hat{\boldsymbol{\alpha}}_{i,t}\|^2$, and calculates the gradient $\nabla_t \triangleq \nabla_{\hat{\boldsymbol{\alpha}}_t} l_t(\hat{\boldsymbol{\alpha}}_t)$ (lines 8-9). Algorithm 1 then updates the parameter $\hat{\boldsymbol{\alpha}}_t$ to $\hat{\boldsymbol{\alpha}}'_{t+1}$ (line 10) and, finally, projects each $\hat{\boldsymbol{\alpha}}'_{i,t+1}$ back to the domain set $\mathcal{D}$ (line 11).

D. No-Regret Guarantee

We present the sublinear regret bound of Algorithm 1 [17, Theorem 1].

Theorem 1 (Dynamic Regret Guarantee [17]): Assume $\eta = \mathcal{O}(1/\sqrt{T})$ in Algorithm 1. Consider the dynamics in (3) corrupted with unmodeled noise $\boldsymbol{e}_t$. Algorithm 1 achieves

$$\mathrm{Regret}_T^D \leq \mathcal{O}\left(T^{\frac{3}{4}}\right) + \mathcal{O}\left(\sqrt{T\sum_{t=1}^{T}\|\boldsymbol{e}_t\|^2}\right). \tag{8}$$

In the absence of $\boldsymbol{e}_t$, which we consider in the paper, Theorem 1 reduces to $\mathrm{Regret}_T^D \leq \mathcal{O}(T^{\frac{3}{4}})$. Therefore, Theorem 1 serves as a finite-time performance guarantee as well as implies that

TABLE I
PERFORMANCE COMPARISON FOR THE GAZEBO SIMULATIONS IN SECTION V-A. THE TABLE REPORTS THE AVERAGE VALUE OF TRACKING ERROR IN POSITION (cm). THE BLUE NUMBERS CORRESPOND TO THE BETTER OVERALL PERFORMANCE. FAILURE IS DENOTED BY $-$. OUR METHOD ACHIEVES BETTER TRACKING PERFORMANCE THAN NOMINAL MPC.

Terrain	$\boldsymbol{f}_u$ (N)	x			y			z			Overall		
		Nominal	L1	Ours	Nominal	L1	Ours	Nominal	L1	Ours	Nominal	L1	Ours
Flat	$\mathbf{0}$	2.23	2.64	2.70	0.31	0.36	0.32	0.84	0.51	0.51	2.51	2.79	2.85
	$4\boldsymbol{g}$	2.02	2.70	2.89	0.41	0.40	0.41	5.59	1.04	1.02	6.07	3.33	3.39
	$8\boldsymbol{g}$	1.04	2.88	3.06	1.66	0.57	0.60	11.60	2.68	2.10	11.87	4.61	4.30
	$12\boldsymbol{g}$	-	2.99	3.58	-	0.84	0.87	-	4.58	3.00	-	6.17	5.69
Slope	$\mathbf{0}$	2.86	2.78	2.33	0.57	0.52	0.54	1.83	1.85	1.82	3.58	3.50	3.12
	$4\boldsymbol{g}$	3.01	2.73	2.46	0.89	0.72	0.75	7.49	2.85	2.37	8.24	4.29	3.66
	$8\boldsymbol{g}$	1.91	2.65	2.62	1.39	1.09	1.05	13.11	4.09	3.00	13.43	5.40	4.44
	$12\boldsymbol{g}$	–	3.21	3.85	–	2.87	2.99	–	6.96	6.10	–	8.73	8.44
Rough	$\mathbf{0}$	0.93	1.00	0.95	2.32	1.69	1.73	1.30	1.00	1.04	3.14	2.46	2.52
	$2\,[\|\boldsymbol{g}\|,\ 0,\ \|\boldsymbol{g}\|]^\top$	–	0.94	0.98	–	3.38	2.47	–	1.35	1.24	–	4.13	3.26
	$4\boldsymbol{g}$	0.91	0.95	0.96	2.20	2.97	2.47	5.35	1.67	1.62	6.10	3.93	3.45

Algorithm 1 converges to the optimal (non-causal) control policy since $\lim_{T\to\infty} \text{Regret}_T^D / T \to 0$.

The bound holds under the assumptions of stability of the estimated systems, Lipschitzness of $c_t(\boldsymbol{x}, \boldsymbol{u})$ in $\boldsymbol{x}$ and $\boldsymbol{u}$, Lipschitzness of $\hat{\boldsymbol{h}}(\cdot)$ in $\hat{\boldsymbol{\alpha}}$, and $\boldsymbol{h}(\cdot)$ can be expressed as $\frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\Phi}(\cdot, \boldsymbol{\theta}_i)\boldsymbol{\alpha}_i$. We refer readers to [17] for detailed statements of the assumptions.

V. NUMERICAL EXPERIMENTS

We evaluate Algorithm 1 in simulated scenarios of legged control under uncertainty, where the quadruped aims to track a reference trajectory despite unknown external disturbances. We conduct experiments on Gazabo (Section V-A) and MuJoCo (Section V-B) simulators.

A. Gazebo Simulations

Simulation Setup: We employ the Unitree Go2 Robot in https://github.com/robomechanics/quad-sdk Quad-SDK [43] based on Gazebo. The MPC runs at 200 Hz. It takes the contact scheduling, desired foothold positions, and reference trajectory as input, and outputs the desired ground reaction forces to the low-level leg controller. The low-level leg controller runs at 500 Hz.

Control Design: The MPC uses look-ahead horizon $N = 20$ simulating the dynamics for $0.6s$. We use quadratic cost functions with $\boldsymbol{Q} = \text{diag}(\boldsymbol{Q_p}, \boldsymbol{Q_\theta}, \boldsymbol{Q_v}, \boldsymbol{Q_\omega})$, $\boldsymbol{Q_p} = 12.5\mathbf{I}_3$, $\boldsymbol{Q_\theta} = \text{diag}([0.5,\ 0.5,\ 2.5])$, $\boldsymbol{Q_v} = \text{diag}([0.2,\ 0.2,\ 0.4])$, $\boldsymbol{Q_\omega} = \text{diag}([0.1,\ 0.1,\ 0.4])$, and $\boldsymbol{R} = 5e^{-5}\mathbf{I}_{12}$. We use the forward Euler method for discretization. We use as the feature $\boldsymbol{z}_t$ the $\boldsymbol{v}_t$, $\boldsymbol{\theta}_t$, $\boldsymbol{\omega}_t$, and $\boldsymbol{J}_t^\top \boldsymbol{u}_t$. We sample $\boldsymbol{w}_i$ from a Gaussian distribution with standard deviation 0.01. We use $M = 50$ random Fourier features and $\eta = 0.003$, and initialize $\hat{\boldsymbol{\alpha}}$ as a zero vector. We do not specify B_h for the domain set $\mathcal{D}$, and the projection step is not applied. The nonlinear program in (4) is constructed by CasADi [44] and solved by IPOPT [45].

Benchmark Experiment Setup: We consider three types of terrains: flat terrain, slope terrain with 20° inclination, and rough terrain with 0.25 m height variation, shown in Fig. 1 (third & fourth). The quadruped is tasked to walk from position $[0,\ 0]$ to $[6,\ 0]$ while maintaining height of 0.3 m above the ground. In flat and slope terrains, the quadruped walks at 0.75 m/s with $\boldsymbol{f}_u = \mathbf{0}, 4\boldsymbol{g}, 8\boldsymbol{g}$, and $12\boldsymbol{g}$. In rough terrain, the quadruped walks at 0.5 m/s with $\boldsymbol{f}_u = \mathbf{0}$, $2[\|\boldsymbol{g}\|,\ 0\ , \|\boldsymbol{g}\|]^\top$, and $4\boldsymbol{g}$. We use the tracking error in position as the performance metric.

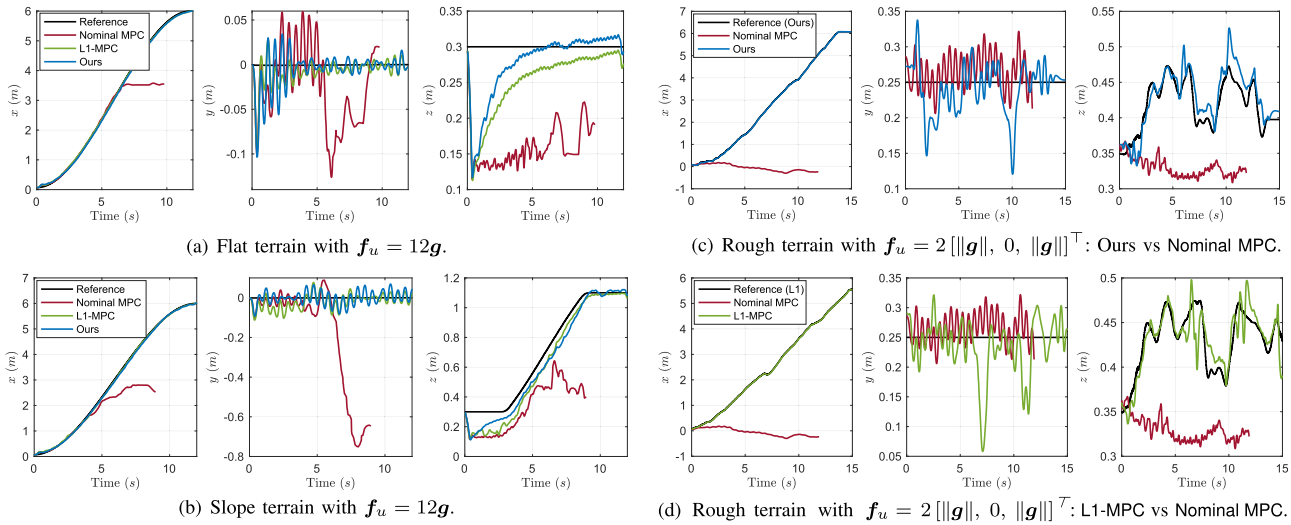
Results: We compare Algorithm 1 with a nominal MPC that assumes no uncertainty in the model, and a heuristic L1 MPC (L1-MPC) based on [46]. Specifically, at each time step, L1-MPC first uses L1 adaptation law to estimate a vector value of $\bar{\boldsymbol{h}}$ Algorithm 1[46]. Then, L1-MPC solves (4) by setting $\hat{\boldsymbol{h}}(\boldsymbol{z}_k) = \bar{\boldsymbol{h}}$ for $k \in \{t, \ldots, t+N-1\}$. We choose L1 adaptation for comparison since it has been successfully applied to quadrotors [46] and quadrupeds [15] recently for online adaptation.

The results are given in Table I and Fig. 3. In Table I, all algorithms perform similarly when $\boldsymbol{f}_u = \mathbf{0}$, as the nominal model is sufficient to capture the quadruped dynamics. Across the scenarios when $\boldsymbol{f}_u$ is non-zero, Algorithm 1 demonstrates significant improvement over the nominal MPC in terms of overall tracking error. Specifically, Algorithm 1 achieves 67% improvement in the case of slope terrain with $\boldsymbol{f}_u = 8\boldsymbol{g}$. Compared to L1-MPC, Algorithm 1 achieves better tracking performance as terrain and external forces become complicated, demonstrating the benefit of learning a model instead of using a vector-value in MPC. Specifically, Algorithm 1 achieves 21% improvement in the case of rought terrain with $\boldsymbol{f}_u = 2[\|\boldsymbol{g}\|,\ 0,\ \|\boldsymbol{g}\|]^\top$. In the case of slope terrain with $\boldsymbol{f}_u = 12\boldsymbol{g}$, we observe that both algorithms perform similarly as the quadruped reaches its limit of handling uncertainty.

In addition, Algorithm 1 enables the quadruped to reach the goal position while the nominal MPC fails, shown in Fig. 3. In flat and slope terrains with $\boldsymbol{f}_u =$, the nominal MPC fails due to the heavy load. In rough terrain with $\boldsymbol{f}_u = 2[\|\boldsymbol{g}\|,\ 0,\ \|\boldsymbol{g}\|]^\top$, the nominal MPC fails to move forward due to the x-component of $\boldsymbol{f}_u$. Compared to L1-MPC, Algorithm 1 exhibits faster response to $\boldsymbol{f_u}$, *e.g.*, z-direction tracking in flat and slope terrains (Fig. 3(a) & (b)) and x-direction tracking in rough terrain (Fig. 3(c) & (d)). Despite being given the same velocity command, our method enables the quadruped to walk faster in the x-direction.

Fig. 4 shows the learned residual forces and torques $\hat{\boldsymbol{h}}(\boldsymbol{z}_t; \hat{\boldsymbol{\alpha}}_t)$ over flat terrain with different $\boldsymbol{f}_u$: (i) constant $\boldsymbol{f}_u = \mathbf{0}, 4\boldsymbol{g}, 8\boldsymbol{g}$, and $12\boldsymbol{g}$, and (ii) time-varying $\boldsymbol{f}_u$ which switches from $6\boldsymbol{g}$ to $12\boldsymbol{g}$ when x-position reaches 3 m. As expected, the main residual dynamics come from the force in z-direction, to which $\hat{\boldsymbol{h}}(\boldsymbol{z}_t; \hat{\boldsymbol{\alpha}}_t)$ converges as the online learning module collects more data on-the-fly. This demonstrates that the online learning module

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) Flat terrain with $\boldsymbol{f}_u = 12\boldsymbol{g}$.

(c) Rough terrain with $\boldsymbol{f}_u = 2\,[\|\boldsymbol{g}\|,\ 0,\ \|\boldsymbol{g}\|]^\top$: Ours vs Nominal MPC.

(b) Slope terrain with $\boldsymbol{f}_u = 12\boldsymbol{g}$.

(d) Rough terrain with $\boldsymbol{f}_u = 2\,[\|\boldsymbol{g}\|,\ 0,\ \|\boldsymbol{g}\|]^\top$: L1-MPC vs Nominal MPC.

Fig. 3. Sample trajectories of the Gazebo Simulations in Section V-A. Three scenarios in flat, slope, and rough terrains are provided. Our method enables the quadruped to reach the goal position while the nominal MPC fails. We plot our method and L1-MPC separately in the case of rough terrain since they move forward at different speeds due to x-direction forces and results in different reference trajectories in x and z. (a) and (b): The nominal MPC fails due to the heavy load. (c) and (d): The nominal MPC fails to move forward due to the x-component of $\boldsymbol{f}_u$. Our method exhibits faster response to $\boldsymbol{f}_u$ than L1-MPC. (a) and (b): faster tracking in z-direction, (c) and (d): faster tracking in x-direction despite the same velocity command, *i.e.*, ours reaches $x = 6$ m faster.

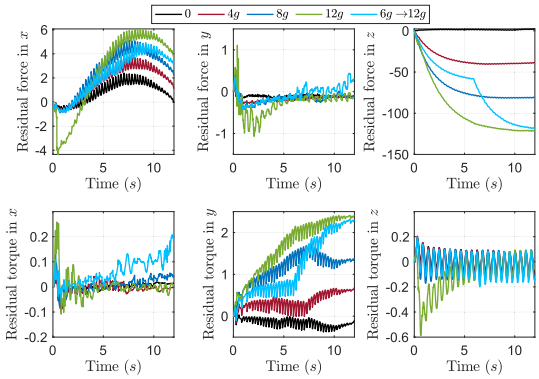


Fig. 4. Learned residual forces and torques $\hat{\boldsymbol{h}}(\boldsymbol{z}_t; \hat{\boldsymbol{\alpha}}_t)$ over flat terrain with different $\boldsymbol{f}_u$ in Section V-A. The main residual dynamics come from the force in z-direction, to which $\hat{\boldsymbol{h}}(\boldsymbol{z}_t; \hat{\boldsymbol{\alpha}}_t)$ converges as the online learning module collects more data on-the-fly. This demonstrates that the online learning module is able to adapt to both constant and time-varying residual dynamics.

is able to adapt to both constant and time-varying residual dynamics.

B. MuJoCo Simulations

Simulation Setup: We evaluate Algorithm 1 on a *sim2sim* setting. Specifically, we employ the controller used in Section V-A to control the Unitree Go2 Robot in https://github.com/unitreerobotics/unitree_mujoco Unitree mujoco based on high-fidelity physics engine DeepMind MuJoCo [47]. The control frequency and parameters are kept the same as in Section V-A.

Benchmark Experiment Setup: We consider flat terrains with constant and varying friction conditions, shown in Fig. 1 (first & second). In the case of varying friction coefficients, the ground coefficients switch between $[0.5, 0.5, 0.01]$ (red rectangle) and $[0.05, 0.05, 0.001]$ (blue rectangle), which stands for sliding, torsional, and rolling frictions, respectively. The quadruped is tasked to walk at $v_x = 0.5$ m/s while maintaining its body height at 0.3 m above the ground. In both terrains, the quadruped carries either no payload or a payload weight at 0 kg, 4 kg, or 8 kg. The 4 kg payload has inertia $[0.00234, 0.00304, 0.00414]$ kg $\cdot$ m^2 and the 8 kg has $[0.00503, 0.00655, 0.00889]$ kg $\cdot$ m^2. The payload and varying ground friction coefficients create time-varying disturbances in both forces and torques: (i) payloads with mass and inertia that create state-dependent forces and torques, and (ii) varying ground friction coefficients that affect the ground reaction forces, therefore creating both force and torque disturbances. Note that under the scenario of no payload, the nominal model in MPC used in Section V-A has around 0.8 kg mismatch to the model simulated in MuJoCo, due to different weights of knee motors. We use the tracking error as the performance metric.

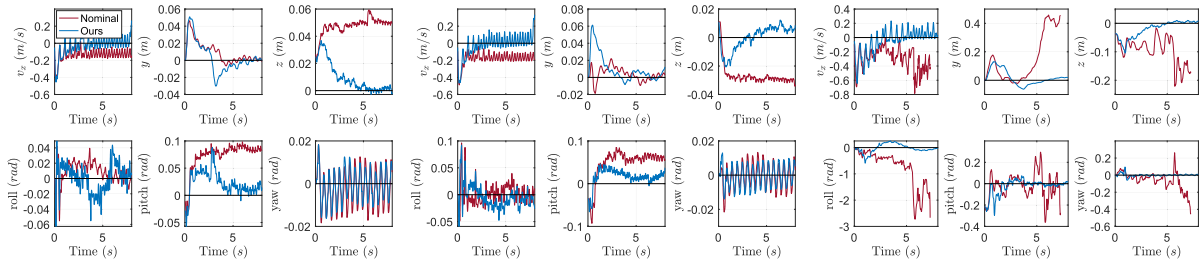
Results: The results are given in Fig. 5 (constant friction) and Fig. 6 (varying friction). In both terrains, our method enables the quadruped to track the 0.5 m/s velocity command while maintaining the 0.3 m body height under different payload conditions. While the Nominal MPC has larger tracking errors in z and v_x, and fails under 8 kg payload.

C. Failure Cases

Despite the online learning module for adapting to the residual dynamics, the method with L1 or random Fourier features may still fail under extreme conditions, such as over $4g$ in rough terrain in the Gazebo simulations and 8 kg in changing friction coefficients in MuJoCo simulations. In those cases, the difficulty of stable walking or learning residual comes from sudden changes in foothold positions or unexpected contact.

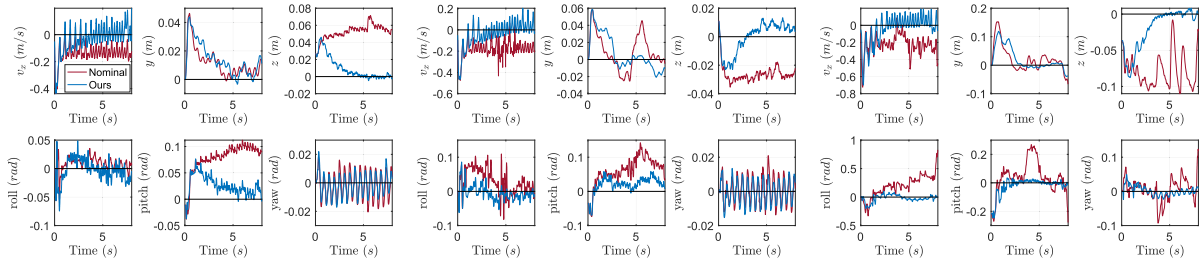
D. Discussion on Real-Time Computation

All the experiments are run on a computer with i7-13700k and 32 GB RAM. Per Algorithm 1, updates of $\boldsymbol{\alpha}$ are carried out at every control cycle; that is, the gradient descent step is executed every time before the MPC is solved. In our implementation, this requires calling the nominal model and using the odometry



(a) No payload. (b) 4 kg payload. (c) 8 kg payload.

Fig. 5. Sample trajectories of the MuJoCo Experiments with constant friction coefficient in Section V-B. Scenarios with no payload, 4 kg, and 8 kg payload are provided. Our method enables the quadruped to track the 0.5 m/s velocity command while maintaining the 0.3 m body height under different payload conditions. While the Nominal MPC has larger tracking errors in z and v_x, and fails under 8 kg payload.



(a) No payload. (b) 4 kg payload. (c) 8 kg payload.

Fig. 6. Sample trajectories of the MuJoCo Experiments with varying friction coefficients in Section V-B. Scenarios with no payload, 4 kg, and 8 kg payload are provided. Our method enables the quadruped to track the 0.5 m/s velocity command while maintaining the 0.3 m body height under different payload conditions. While the Nominal MPC has larger tracking errors in z and v_x, and fails under 8 kg payload.

information to obtain the "ground truth" disturbance, then using the learned residual dynamics to obtain the estimation loss. The gradient for parameter update can be obtained analytically from $l_t(\hat{\boldsymbol{\alpha}}_t) \triangleq \|\boldsymbol{h}(\boldsymbol{z}_t) - \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\Phi}(\boldsymbol{z}_t, \boldsymbol{\theta}_i)\hat{\boldsymbol{\alpha}}_{i,t}\|^2$ due to the linearity of $\hat{\boldsymbol{h}}$ in $\hat{\boldsymbol{\alpha}}$. Therefore, the online update process is a lightweight add-on to the original MPC loop. On the other hand, the residual dynamics $\hat{\boldsymbol{h}}$ add complexity to the nominal model, and the MPC solve time will increase as M increases. In our experiments, we use $M = 50$ so that the control frequency remains 200 Hz. The complexity of the nominal dynamics also affects the real-time computation. Residual dynamics in joint space can be captured by incorporating joint dynamics into nominal dynamics, but this increases the state dimension and introduces additional computational burden.

VI. Conclusion

We provided Algorithm 1 for *Adaptive Legged Locomotion via Online Learning and Model Predictive Control* (Problem 1). The algorithm is composed of two interacting components: MPC and online learning of residual dynamics. The algorithm uses random Fourier features to approximate the residual dynamics in reproducing kernel Hilbert spaces. Then, it employs MPC based on the current learned model of the residual dynamics. The model of the residual dynamics is updated online in a self-supervised manner using least squares based on the data collected while controlling the quadruped. Algorithm 1 guarantees no dynamic regret against an optimal clairvoyant (non-causal) policy that knows the residual dynamics a priori (Theorem 1). The proposed Algorithm 1 is validated in the simulation environment with high-fidelity physics engines. Our simulations include quadruped aiming to track a reference trajectory despite constant uncertainty up to $12\boldsymbol{g}$ in flat, slope, and rough terrains. The algorithm (i) achieves up to 67% improvement of tracking performance over the nominal MPC and 21% improvement over L1-MPC, and (ii) succeeds even when nominal MPC fails. We also validate Algorithm 1 under time-varying uncertainty in flat terrains of different coefficients with up to 8 kg payload, showing the algorithm achieves significantly better tracking performance than Nominal MPC.

References

[1] D. Seneviratne et al., "Smart maintenance and inspection of linear assets: An industry 4.0 approach," *Acta Imeko*, vol. 7, pp. 50–56, 2018.
[2] J. Tan et al., "Sim-to-real: Learning agile locomotion for quadruped robots," in *Proc. Robot.: Sci. Syst.*, Pittsburgh, PA, USA, Jun. 2018, doi: 10.15607/RSS.2018.XIV.010.
[3] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau5872.
[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, 2020, Art. no. eabc5986.
[5] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Proc. Conf. Robot Learn.*, 2022, pp. 22–31.
[6] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 2908–2927, Oct. 2022.
[7] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Proc. Robot.: Sci. Syst.*, Jul. 2021, doi: 10.15607/RSS.2021.XVII.011.

[8] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4630–4637, Apr. 2022.

[9] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 5078–5084.

[10] Y. Zhong, C. Zhang, T. He, and G. Shi, "Bridging adaptivity and safety: Learning agile collision-free locomotion across varied physics," in *Proc. 7th Annu. Learn. Dyn. Control Conf.*, 2025, pp. 1498–1511.

[11] A. Pandala, R. T. Fawcett, U. Rosolia, A. D. Ames, and K. A. Hamed, "Robust predictive control for quadrupedal locomotion: Learning to close the gap between reduced-and full-order models," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 6622–6629, Jul. 2022.

[12] S. Xu, L. Zhu, H.-T. Zhang, and C. P. Ho, "Robust convex model predictive control for quadruped locomotion under uncertainties," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4837–4854, Dec. 2023.

[13] M. V. Minniti, R. Grandia, F. Farshidian, and M. Hutter, "Adaptive clf-mpc with application to quadrupedal robots," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 565–572, Jan. 2022.

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

[15] M. Sombolestan and Q. Nguyen, "Adaptive force-based control of dynamic legged locomotion over uneven terrain," *IEEE Trans. Robot.*, vol. 40, pp. 2462–2477, 2024.

[16] M. Elobaid et al., "Adaptive non-linear centroidal mpc with stability guarantees for robust locomotion of legged robots," *IEEE Robot. Automat. Lett.*, vol. 10, no. 3, pp. 2806–2813, Mar. 2025.

[17] H. Zhou and V. Tzoumas, "Simultaneous system identification and model predictive control with no dynamic regret," *IEEE Trans. Robot.*, vol. 41, pp. 4322–4341, 2025.

[18] E. Hazan et al., "Introduction to online convex optimization," *Foundations Trends Optim.*, vol. 2, no. 3-4, pp. 157–325, 2016.

[19] F. Cucker and S. Smale, "On the mathematical foundations of learning," *Bull. Amer. Math. Soc.*, vol. 39, no. 1, pp. 1–49, 2002.

[20] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inform. Process. Syst.*, 2007, vol. 20.

[21] A. Rahimi and B. Recht, "Uniform approximation of functions with random bases," in *Proc. 2008 46 th Annu. Allerton Conf. Commun., Control, Comput.*, 2008, pp. 555–561.

[22] N. M. Boffi, S. Tu, and J.-J. E. Slotine, "Nonparametric adaptive control and prediction: Theory and randomized algorithms," *J. Mach. Learn. Res.*, vol. 23, no. 281, pp. 1–46, 2022.

[23] J. Luo and K. Hauser, "Robust trajectory optimization under frictional contact with iterative learning," *Auton. Robots*, vol. 41, pp. 1447–1461, 2017.

[24] D. Q. Mayne, E. C. Kerrigan, E. V. Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *Int. J. Robust Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[25] D. M. Raimondo, D. Limon, M. Lazar, L. Magni, and E. F. N. Camacho, "Min-max model predictive control of nonlinear systems: A unifying overview on stability," *Eur. J. Control*, vol. 15, no. 1, pp. 5–21, 2009.

[26] J.-J. E. Slotine, "Applied nonlinear control," *PRENTICE-HALL Google Schola*, vol. 2, pp. 1123–1131, 1991.

[27] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos, *Nonlinear and adaptive control design.* Hoboken, NJ, USA: Wiley, 1995.

[28] P. A. Ioannou and J. Sun, *Robust Adaptive Control*, vol. 1. Englewood Cliffs, NJ, USA: Prentice-Hall, 1996.

[29] N. Agarwal, B. Bullins, E. Hazan, S. Kakade, and K. Singh, "Online control with adversarial disturbances," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 111–119.

[30] H. Zhou and V. Tzoumas, "Safe non-stochastic control of linear dynamical systems," in *Proc. 2023 62nd IEEE Conf. Decis. Control*, 2023, pp. 5033–5038.

[31] H. Zhou, Z. Xu, and V. Tzoumas, "Efficient online learning with memory via frank-wolfe optimization: Algorithms with bounded dynamic regret and applications to control," in *Proc. 2023 62nd IEEE Conf. Decis. Control*, 2023, pp. 8266–8273.

[32] H. Zhou, Y. Song, and V. Tzoumas, "Safe non-stochastic control of control-affine systems: An online convex optimization approach," *IEEE Robot. Automat. Lett.*, vol. 8, no. 12, pp. 7873–7880, Dec. 2023.

[33] N. M. Boffi, S. Tu, and J.-J. E. Slotine, "Regret bounds for adaptive nonlinear control," in *Proc. Learn. Dyn. Control.*, 2021, pp. 471–483.

[34] A. Tsiamis, A. Karapetyan, Y. Li, E. C. Balta, and J. Lygeros, "Predictive linear online tracking for unknown targets," in *Proc. 41st Int. Conf. Mach. Learn.*, 2024, pp. 48657–48 694.

[35] H. Zhou and V. Tzoumas, "No-regret model predictive control with online learning of Koopman operators," in *Proc. 2025 Amer. Control Conf.*, 2025.

[36] R. Brault, M. Heinonen, and F. Buc, "Random fourier features for operator-valued kernels," in *Proc. Asian Conf. Mach. Learn.*, 2016, pp. 110–125.

[37] H. Q. Minh, "Operator-valued bochner theorem, fourier feature maps for operator-valued kernels, and vector-valued learning," 2016, *arXiv:1608.05639*.

[38] F. Bach, "Breaking the curse of dimensionality with convex neural networks," *J. Mach. Learn. Res.*, vol. 18, no. 19, pp. 1–53, 2017.

[39] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, "Linearized two-layers neural networks in high dimension," *Ann. Stat.*, vol. 49, no. 2, pp. 1029–1054, 2021.

[40] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. Adv. Neural Inform. Process. Syst.*, 2018, vol. 31.

[41] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robot. Automat. Lett.*, vol. 8, no. 4, pp. 2397–2404, Apr. 2023.

[42] A. Saviolo, J. Frey, A. Rathod, M. Diehl, and G. Loianno, "Active learning of discrete-time dynamics for uncertainty-aware model predictive control," *IEEE Trans. Robot.*, vol. 40, pp. 1273–1291, 2024.

[43] J. Norby et al., "Quad-SDK: Full stack software framework for agile quadrupedal locomotion," in *Proc. ICRA Workshop Legged Robots*, 2022, pp. 1–5.

[44] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, pp. 1–36, 2019.

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

[46] Z. Wu et al., "L1quad: L1 adaptive augmentation of geometric control for agile quadrotors with performance guarantees," *IEEE Trans. Control Syst. Technol.*, vol. 33, no. 2, pp. 597–612, Mar. 2025.

[47] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Proc. 2012 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.