Tuning Legged Locomotion Controllers via Safe Bayesian Optimization

Anonymous Author(s)
Affiliation
Address
`email`

Abstract:
In this paper, we present a data-driven strategy to simplify the deployment of model-based controllers in legged robotic hardware platforms. Our approach leverages a model-free safe learning algorithm to automate the tuning of control gains, addressing the mismatch between the simplified model used in the control formulation and the real system. This method substantially mitigates the risk of hazardous interactions with the robot by sample-efficiently optimizing parameters within a probably safe region. Additionally, we extend the applicability of our approach to incorporate the different gait parameters as contexts, leading to a safe, sample-efficient exploration algorithm capable of tuning a motion controller for diverse gait patterns. We validate our method through simulation and hardware experiments, where we demonstrate that the algorithm obtains superior performance on tuning a model-based motion controller for multiple gaits safely.

1 Introduction

A model-based control approach can produce highly dynamic and diverse motions for legged robots. This strategy enables rapid adjustment to different robots and bypasses the need for offline training, consequently accelerating the design and testing stages. Nonetheless, it requires an accurate dynamics model of the system, which is often unavailable due to our limited understanding of real-world physics and inevitable simplifications to reduce the computational burden. Consequently, these controllers typically underperform when applied directly to hardware, requiring significant parameter fine-tuning. This tuning process is time-consuming and can also harm the hardware platform. Furthermore, it often needs repetition for different environments or motion patterns to ensure consistent performance across a variety of settings.

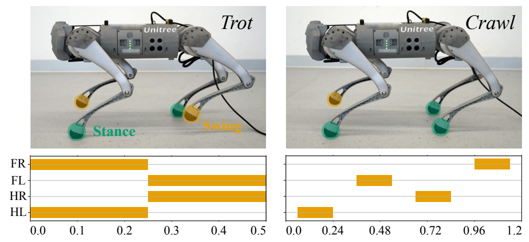

Figure 1: Snapshots of the *Unitree Go1* robot performing *Trot* (left) and *crawl* gaits (right).

This work explores the challenge of identifying optimal control gain parameters for a model-based controller that improves the robustness and tracking performance by bridging the gap between simplified models and real-world dynamics. For this purpose, we utilize GoSafeOpt [1] to automate the parameter tuning process, enabling the online identification of optimal control gain parameters within a safe region, efficiently utilizing samples and thus safeguarding the hardware platforms during optimization. Furthermore, we extend GoSafeOpt to incorporate the different gait parameters as contexts. This enables sample-efficient learning of control gains across different gaits. For the resulting contextual GoSafeOpt algorithm, we give theoretical safety and optimality guarantees.

We demonstrate our method on the quadruped robot *Unitree Go1* [2] in both simulation and hardware experiments. This corresponds to a six dimensional tuning task with a five dimensional context. In the simulation, we show that contextual GoSafeOpt outperforms other model-free safe exploration baselines while making *no unsafe* interactions. Moreover, when trained for different gait patterns, our results clearly indicate that including the gaits as contexts, results in a considerable performance boost. In our hardware experiments, we show that contextual GoSafeOpt finds optimal feedback controller gains for both the trot and crawl gaits in only 50 learning steps, each while having *no unsafe interaction with the real robot*.

In summary, the main contributions of this work are as follows; (*i*) we formulate the controller parameter tuning problem as a constrained optimization and integrate the different gait patterns as context variables [3], (*ii*) we extend GoSafeOpt [1] to the contextual case for which we give safety and optimality guarantees. Furthermore, (*iii*) we demonstrate the superiority of contextual GoSafeOpt over other state-of-the-art safe exploration algorithms in simulation, and (*iv*) we show that GoSafeOpt successfully and safely tunes feedback control policies over two different gaits directly on the hardware. To the best of our knowledge, we are the first to extend GoSafeOpt to the contextual case and to apply it to a highly dynamic and complex real-world system like the *Unitree Go1*.

2 Related Work

Bridging the reality gap in quadrupedal locomotion tasks Several previous studies have emphasized the importance of considering an actuator behavior and identifying the system latency to successfully bridge the *reality gap* in legged robot systems [4, 5, 6]. These studies develop a simulation model of a legged robot system, incorporate either modeled or learned actuator dynamics and train a control policy that can be effectively deployed to the robot hardware.

Incorporating this strategy into a model-based control framework is an area of active investigation. In a model-based control regime, it's typically more straightforward to introduce adjustable control gain parameters and fine-tune them to align with the real-world behaviors of the robot. For instance, Kim et al. [7], Kang et al. [8] use joint position- and velocity-level feedback to joint torque command in order to address any discrepancy between the actual torque output and the intended torque command for robots with proprioceptive actuators [9]. However, the fine-tuning of these parameters continues to present a significant challenge. Schperberg et al. [10] utilize the unscented Kalman filter algorithm to recursively tune control parameters of a model-based motion controller online, and they successfully demonstrate it on the simulated quadrupedal robot in the presence of sensor noise and joint-level friction. However, their proposed tuning method is inherently unsafe and can therefore lead to arbitrary harmful interactions with the system. In contrast, our method is evaluated on hardware and generalizes to multiple gaits while avoiding any unsafe interactions with the robot.

Safe exploration for controller parameter tuning Training a controller directly on hardware is a challenging task, as it requires sample efficient and safe exploration to avoid possible damage to the robot. In such a context, Bayesian optimization (BO [11]) emerges as a suitable framework due to its sample efficiency. A notable example in the field of legged robotics comes from Calandra et al. [12], who successfully employed BO to learn optimal gait parameters for a bipedal robot platform using data obtained from hardware experiments.

BO methods can be easily adapted to constrained settings for safe learning. Gelbart et al. [13], Hernández-Lobato et al. [14], Marco et al. [15] utilize constrained BO for finding safe optimal controller parameters. However, these works do not provide safety assurance during exploration. In contrast, methods such as SafeOpt [16, 17] and its extensions [18, 19, 20, 1] guarantee safety throughout the entire learning and exploration phases. Starting from an initial safe controller, SafeOpt leverages regularity properties of the underlying optimization to expand the set of safe controllers. This expansion is inherently local, and accordingly SafeOpt can miss the global optimum. For dynamical systems, Baumann et al. [20] propose GoSafe, a *global* safe exploration algorithm. GoSafe alternates between local safe exploration, where it also learns safe backup policies, and global exploration, where it uses the learned backup policies to guarantee safety. Therefore, whereas SafeOpt might be restricted to a local optimum, GoSafe can find the global one. However, the BO routine proposed in GoSafe is expensive and sample inefficient for all but low dimensional systems [1]. To this end, Sukhija et al. [1] introduce GoSafeOpt. GoSafeOpt leverages the underlying Markovian structure of the dynamical system to overcome GoSafe's restrictions. As a result, GoSafeOpt can perform global safe exploration for realistic and high-dimensional dynamical systems.

We extend GoSafeOpt by incorporating a contextual setting and apply it to a quadruped robot. In this application, we optimize the controller parameters for various gait patterns. It's important to highlight that this domain is considerably high-dimensional, involving a twenty-four-dimensional state space, six-dimensional parameter space, and five-dimensional context space.

3 Problem Setting

Safe learning formulation The dynamics of robotic systems can generally be described as an ordinary differential equation (ODE) of the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ where $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^{d_u}$ is the control signal and $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^{d_x}$ is the generalized robot state. Due to the reality gap, disparities can arise between the real-world dynamics and the dynamics model $\boldsymbol{f}$. This often results in a significant divergence between the behaviors of models and actual real-world systems, thereby making the control of intricate and highly dynamic entities like quadrupeds particularly challenging.

A common solution to this problem is using a feedback policy to rectify the model inaccuracies. Given a desired input signal $\boldsymbol{u}^*$, desired state $\boldsymbol{x}^*$, and true system state $\boldsymbol{x}$, we formulate a parameterized feedback controller in the form $\boldsymbol{u} = \pi_{\boldsymbol{\theta}}(\boldsymbol{u}^*, \boldsymbol{x}^*, \boldsymbol{x})$ that steers $\boldsymbol{x}$ to closely align with $\boldsymbol{x}^*$. The parameters $\boldsymbol{\theta}$ are picked to minimize the tracking error. A common example of such a feedback policy is PD control, where $\boldsymbol{u} = \boldsymbol{u}^* + \boldsymbol{\theta}[(\boldsymbol{x}^* - \boldsymbol{x})^\top, (\dot{\boldsymbol{x}}^* - \dot{\boldsymbol{x}})^\top]^\top$, where $\boldsymbol{\theta} \in \mathbb{R}^{d_u \times 2d_x}$ corresponds to the controller gains. Typically, choosing the parameters $\boldsymbol{\theta}$ involves a heuristic process, requiring experimental iterations with the physical hardware. However, such interactions can be unpredictably risky and could possibly cause damage to the hardware.

In this work, we formalize the tuning process as a constrained optimization problem:

$$\max_{\boldsymbol{\theta} \in \Theta} g(\boldsymbol{\theta}) \quad \text{such that } q_i(\boldsymbol{\theta}) \geq 0, \forall i \in \mathcal{I}_q, \tag{1}$$

where g is an objective function, q_i are the constraints with $\mathcal{I}_q = \{1, \ldots, c\}$, and Θ is a compact set of parameters over which we optimize. Since the true dynamics are unknown, we cannot solve Equation (1) directly. Instead, we interact with the robot to learn $g(\boldsymbol{\theta})$ and $q_i(\boldsymbol{\theta})$, and solve the optimization problem in a black-box fashion. As we interact directly with the robot hardware, it is important that the learning process is sample-efficient and safe, i.e., constraints q_i are not violated during learning. To this end, we use the model-free safe learning algorithm GoSafeOpt.

Extension to multiple gait patterns We expand the applicability of our method to support a range of quadrupedal gait patterns and enable smooth online transitions among them. Each specific gait pattern demonstrates unique dynamic properties, therefore, the optimal feedback parameters $\boldsymbol{\theta}$ vary depending on the gait pattern in question. We consider gaits as *contexts* $\boldsymbol{z}$ from a (not necessarily finite) set of contexts $\mathcal{Z}$ [3]. Contexts are essentially external variables specified by the user and remain untouched by the optimization process. We broaden our initial problem formulation from Equation (1) to include these contexts;

$$\max_{\boldsymbol{\theta} \in \Theta} g(\boldsymbol{\theta}, \boldsymbol{z}) \quad \text{such that } q_i(\boldsymbol{\theta}, \boldsymbol{z}) \geq 0, \forall i \in \mathcal{I}_q, \tag{2}$$

where $\boldsymbol{z} \in \mathcal{Z}$ is the context, which in our scenario, is the gait pattern of interest.

It is noteworthy that the prior work by Sukhija et al. [1] only introduces GoSafeOpt for the non-contextual setting. In this work, we extend it to the contextual case and give safety as well as optimality guarantees.

3.1 Assumptions

In this section, we reiterate the assumptions from Sukhija et al. [1] for GoSafeOpt.

Assumption 1 (Initial safe seed). *For any episode $n \geq 1$ with (user-specified) context $\boldsymbol{z}_n \in \mathcal{Z}$, a non-empty initial safe set of parameters $\mathcal{S}_{n-1}(\boldsymbol{z}_n) \subset \Theta$ is known. That is, for all $\boldsymbol{\theta} \in \mathcal{S}_{n-1}(\boldsymbol{z}_n)$ and all $i \in \mathcal{I}_q$, $q_i(\boldsymbol{\theta}, \boldsymbol{z}_n) \geq 0$.*

Here, $\mathcal{S}_n(\boldsymbol{z}) \supseteq \mathcal{S}_0(\boldsymbol{z})$ denotes the safe set after episode n for the given context $\boldsymbol{z}$ as defined in Equation (14) in Appendix A. Given the prior knowledge of the dynamics, a conservative safe set of parameters represents some initial stable feedback controller. Accordingly, this assumption is typically satisfied in practice. The assumption is necessary as, in principle, during each iteration, an adversarial context could be chosen for which the initial safe set does not include any safe parameters.

Assumption 2 (Continuity of objective and constraints)**.** *Let h be defined as*

$$h(\boldsymbol{\theta}, \boldsymbol{z}, i) = \begin{cases} g(\boldsymbol{\theta}, \boldsymbol{z}) & \text{if } i = 0, \\ q_i(\boldsymbol{\theta}, \boldsymbol{z}) & \text{if } i \in \mathcal{I}_q. \end{cases} \tag{3}$$

We assume that h lies in a reproducing kernel Hilbert space (RKHS) associated with a kernel k and has a bounded norm in that RKHS, that is, $\|h\|_k \leq B$. Furthermore, we assume that g and q_i $(\forall i \in \mathcal{I}_q)$ are Lipschitz-continuous with known Lipschitz constants.

This is a common assumption in the model-free safe exploration literature [17, 20, 1]. Sukhija et al. [1] discuss the practical implications of this assumption in more detail.

Assumption 3. *We obtain noisy measurements of h with measurement noise i.i.d. σ-sub-Gaussian. Specifically, for a measurement y_i of $h(\boldsymbol{\theta}, \boldsymbol{z}, i)$, we have $y_i = h(\boldsymbol{\theta}, \boldsymbol{z}, i) + \epsilon_i$ with ϵ_i σ-sub-Gaussian for all $i \in \mathcal{I}$ where we write $\mathcal{I} = \{0, \dots, c\}$.*

Assumption 4. *We observe the state $\boldsymbol{x}(t)$ every Δt seconds. Furthermore, for any $\boldsymbol{x}(t)$ and $\rho \in [0, 1]$, the distance to $\boldsymbol{x}(t + \rho\Delta t)$ induced by any action is bounded by a known constant Ξ, that is, $\|\boldsymbol{x}(t + \rho\Delta t) - \boldsymbol{x}(t)\| \leq \Xi$.*

Assumption 4 is crucial to guarantee safety in continuous time even though the state is measured at discrete time instances. For highly dynamic systems, such as quadrupeds, the observation frequency is typically very high, e.g., $500\,\text{Hz}$ - $1\,\text{kHz}$, and accordingly Ξ is small.

Assumption 5. *We assume that, for all $i \in \{1, \dots, c\}$, q_i is defined as the minimum of a state-dependent function $\bar{q}_i$ along the trajectory starting in $\boldsymbol{x}_0$ with controller $\boldsymbol{\pi}_{\boldsymbol{\theta}}$. Formally,*

$$q_i(\boldsymbol{\theta}, \boldsymbol{z}) = \min_{\boldsymbol{x}' \in \xi_{(\boldsymbol{x}_0, \boldsymbol{\theta}, \boldsymbol{z})}} \bar{q}_i(\boldsymbol{x}', \boldsymbol{z}), \tag{4}$$

with $\xi_{(\boldsymbol{x}_0, \boldsymbol{\theta}, \boldsymbol{z})} = \{\boldsymbol{x}_0 + \int_0^t \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{x}(\tau), \boldsymbol{z}))\, d\tau \mid t \geq 0\}$ representing the trajectory of $\boldsymbol{x}(t)$ under policy parameter $\boldsymbol{\theta}$ and context $\boldsymbol{z}$ starting from $\boldsymbol{x}_0$ at time 0.

Assumption 5 is an assumption on our choice of the constraint. Many common constraints, such as the minimum distance to an obstacle along a trajectory, satisfy this assumption.

4 GoSafeOpt for Controller Optimization for Quadrupedal Locomotion

In this section, we first provide a brief overview of our model-based locomotion controller and the gait parameterization. Following this, we discuss GoSafeOpt and its contextual extension, for which we provide safety and optimality guarantees.

4.1 Control Pipeline

Model-based locomotion controller Our locomotion controller utilizes a combination of the model predictive control (MPC) and the whole-body control (WBC) method following the previous work by Kim et al. [7], Kang et al. [8]. The MPC generates dynamically consistent base and foot trajectories by finding an optimal solution of a finite-horizon optimal control problem, using a simplified model. To convert these trajectories into joint-level control signals, we implement a WBC method that incorporates a more sophisticated dynamics model and takes into account the physical constraints of the robot. More specifically, we use a WBC formulation similar to the one presented by Kim et al. [7]. This method calculates the desired generalized coordinates $\boldsymbol{x}^{\text{cmd}}$, speed $\dot{\boldsymbol{x}}^{\text{cmd}}$, and acceleration $\ddot{\boldsymbol{x}}^{\text{cmd}}$ on a kinematic level while respecting task priority via the null-space projection [21]. Subsequently, it finds joint torque commands by solving a quadratic program that aligns with the desired generalized acceleration, adhering to the motion equations of the floating base and other physical constraints. For a more detailed explanation of the WBC formulation, the reader is referred to Appendix C.

We emphasize that the feed-forward torque commands by themselves fail to produce desired motion on the robot hardware due to model discrepancies. Particularly, we observed the actuator dynamics and joint friction, which are impractical to include in the system model, contribute significantly to this model mismatch. As a practical solution, we compute the final joint torque commands $\boldsymbol{\tau}^{\text{cmd}} = \boldsymbol{\tau} + \boldsymbol{k}_p(\boldsymbol{x}^* - \boldsymbol{x}) + \boldsymbol{k}_d(\dot{\boldsymbol{x}}^* - \dot{\boldsymbol{x}})$ and send them to the robot with the feedback gains $\boldsymbol{k}_p \in \mathbb{R}^{d_u \times d_x}$ and $\boldsymbol{k}_d \in \mathbb{R}^{d_u \times d_x}$.

Gait parameterizations We parameterize a quadrupedal gait pattern with $\boldsymbol{z}^g = [d^g, t_s^g, o_1^g, o_2^g, o_3^g]$, where d^g is the duty cycle for gate g, t_s^g is the stride duration and o_i^g are the offsets of legs two to four respectively. The duty cycle is defined as the contact duration divided by the stride duration. In general, the optimal feedback parameters $(\boldsymbol{k}_p^*, \boldsymbol{k}_d^*)$ change with the gait. We show this empirically in Section 5.

4.2 Contextual GoSafeOpt

We model the unknown objective and constraint functions through Gaussian Process regression [22]. To this end, given a dataset $\{\boldsymbol{v}_j, \boldsymbol{y}_j\}_{j \le n}$, with $\boldsymbol{v}_j = (\boldsymbol{\theta}_j, \boldsymbol{z}_j)$ and the kernel k, we calculate a mean and uncertainty estimate of our function:

$$\begin{aligned} \mu_n(\boldsymbol{v}, i) &= \boldsymbol{k}_n^\top(\boldsymbol{v})(\boldsymbol{K}_n + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}_{n,i}, \\ \sigma_n^2(\boldsymbol{v}, i) &= k(\boldsymbol{v}, \boldsymbol{v}) - \boldsymbol{k}_n^\top(\boldsymbol{v})(\boldsymbol{K}_n + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_n(\boldsymbol{v}), \end{aligned} \tag{5}$$

where $\boldsymbol{y}_{n,i} = [y_{j,i}]_{j \le n}^\top$ are the observations of $h(\cdot, i)$, $\boldsymbol{k}_n(\boldsymbol{v}) = [k(\boldsymbol{v}, \boldsymbol{v}_j)]_{j \le n}^\top$, and $\boldsymbol{K}_n = [k(\boldsymbol{v}_j, \boldsymbol{v}_l)]_{j,l \le n}$ is the kernel matrix. We leverage these estimates to provide high-probability frequentist confidence intervals.

Lemma 1 (Confidence intervals, Theorem 2 of [23] and Lemma 4.1 of [17])**.** *For any $\delta \in (0, 1)$ and under Assumptions 2 and 3, with probability at least $1 - \delta$ it holds jointly for all $n, i, \boldsymbol{z}, \boldsymbol{\theta}$ that*

$$|h(\boldsymbol{\theta}, \boldsymbol{z}, i) - \mu_n(\boldsymbol{\theta}, \boldsymbol{z}, i)| \le \beta_n(\delta) \cdot \sigma_n(\boldsymbol{\theta}, \boldsymbol{z}, i) \tag{6}$$

with $\beta_n(\delta) \le \mathcal{O}(B + 4\sigma\sqrt{2(\gamma_{n|\mathcal{I}|} + 1 + \log(1/\delta))})$ where

$$\gamma_n = \max_{\substack{A \subset \Theta \times \mathcal{Z} \times \mathcal{I} \\ |A| \le n}} \mathrm{I}(\boldsymbol{y}_A; \boldsymbol{h}_A). \tag{7}$$

Here, $\mathrm{I}(\boldsymbol{y}_A; \boldsymbol{h}_A)$ denotes the *mutual information* between $\boldsymbol{h}_A = [h(\boldsymbol{v})]_{\boldsymbol{v} \in A}$, if modeled with a GP, and the noisy observations $\boldsymbol{y}_A$ at $\boldsymbol{h}_A$, and it quantifies the reduction in uncertainty about h upon observing $\boldsymbol{y}_A$ at points A. The quantity γ_n is a Bayesian construct, however, in the frequentist setting it quantifies the complexity of learning the function $\boldsymbol{h}$. It is instance dependent and can be bounded depending on the domain $\Theta \times \mathcal{Z} \times \mathcal{I}$ and kernel function k (see Appendix A).

Given the confidence interval from Equation (6), we define a confidence set for each context $\boldsymbol{z}$, parameter $\boldsymbol{\theta}$ and index $0 \le i \le c$, as

$$C_0(\boldsymbol{\theta}, \boldsymbol{z}, i) = \begin{cases} [0, \infty] & \text{if } \boldsymbol{\theta} \in \mathcal{S}_0(\boldsymbol{z}) \text{ and } i \ge 1, \\ [-\infty, \infty] & \text{otherwise,} \end{cases} \tag{8}$$

$$C_n(\boldsymbol{\theta}, \boldsymbol{z}, i) = C_{n-1}(\boldsymbol{\theta}, \boldsymbol{z}, i) \cap [\mu_n(\boldsymbol{\theta}, \boldsymbol{z}, i) \pm \beta_n(\delta) \cdot \sigma_n(\boldsymbol{\theta}, \boldsymbol{z}, i)], \tag{9}$$

We refer to $l_n(\boldsymbol{\theta}, \boldsymbol{z}, i) = \min C_n(\boldsymbol{\theta}, \boldsymbol{z}, i)$ as the lower bound, $u_n(\boldsymbol{\theta}, \boldsymbol{z}, i) = \max C_n(\boldsymbol{\theta}, \boldsymbol{z}, i)$ the upper bound, and $w_n(\boldsymbol{\theta}, \boldsymbol{z}, i) = u_n(\boldsymbol{\theta}, \boldsymbol{z}, i) - l_n(\boldsymbol{\theta}, \boldsymbol{z}, i)$ the width of our confidence set.

4.2.1 Algorithm

An episode n of contextual GoSafeOpt with the given (user-specified) context $\boldsymbol{z}_n \in \mathcal{Z}$ is performed in one of two alternating stages: *local safe exploration* (LSE) and *global exploration* (GE).

Local safe exploration During the LSE stage, we explore the subset of the parameter space Θ which is known to be safe, and learn backup policies for each visited state. In this stage, the parameters are selected according to the acquisition function

$$\boldsymbol{\theta}_n = \underset{\boldsymbol{\theta} \in \mathcal{G}_{n-1}(\boldsymbol{z}_n) \cup \mathcal{M}_{n-1}(\boldsymbol{z}_n)}{\operatorname{argmax}} \max_{i \in \mathcal{I}} w_{n-1}(\boldsymbol{\theta}, \boldsymbol{z}_n, i) \tag{10}$$

where $\boldsymbol{h}$, $\mathcal{G}_n(\boldsymbol{z}_n) \subseteq S_n(\boldsymbol{z}_n)$ is a set of "expanders" (c.f., Equation (15) in Appendix A) and $\mathcal{M}_n(\boldsymbol{z}_n) \subseteq S_n(\boldsymbol{z}_n)$ is a set of "maximizers" (c.f., Equation (17) in Appendix A). Intuitively, $\mathcal{G}_n(\boldsymbol{z}_n) \cup \mathcal{M}_n(\boldsymbol{z}_n)$ represents those parameters that can potentially lead to an expansion of the safe set $\mathcal{S}_n(\boldsymbol{z}_n)$ or potentially be a solution to the optimization problem of Equation (2) with context $\boldsymbol{z}_n$.

Global exploration If LSE converged (see Equation (20) in Appendix A), we run the GE stage where we evaluate possibly unsafe policies and trigger a backup policy whenever necessary. If no backup policy is triggered, we conclude that the evaluated policy is safe and add it to our safe set. After a new parameter is added to the safe set during GE, we continue with LSE.

The parameters are selected according to the acquisition function

$$\boldsymbol{\theta}_n = \underset{\boldsymbol{\theta} \in \Theta \setminus (S_{n-1}(\boldsymbol{z}_n) \cup \mathcal{E}(\boldsymbol{z}_n))}{\operatorname{argmax}} \max_{i \in \mathcal{I}} w_{n-1}(\boldsymbol{\theta}, \boldsymbol{z}_n, i) \tag{11}$$

where $\mathcal{E}$ denotes all parameters which have been shown to be unsafe (see line 7 of Algorithm 4 in Appendix A). If all parameters have been determined as either safe or unsafe, i.e., $\Theta \setminus (S_n(\boldsymbol{z}_n) \cup \mathcal{E}(\boldsymbol{z}_n)) = \emptyset$, then GE has converged.

Summary A detailed description of the contextual GoSafeOpt algorithm is provided in Appendix A.2. GoSafeOpt alternates between local safe exploration and global exploration. Therefore, it can seek for the optimum globally. We provide an example in Figure 4 in Appendix B.

The only difference between the contextual and non-contextual variants is that contextual GoSafeOpt maintains separate sets $S_n, C_n, \mathcal{B}_n, \mathcal{D}_n, \mathcal{E}$, and $\mathcal{X}_{\text{Fail}}$ for each context $\boldsymbol{z} \in \mathcal{Z}$. For any given context $\boldsymbol{z} \in \mathcal{Z}$, the running best guess of contextual GoSafeOpt for the optimum is $\hat{\boldsymbol{\theta}}_n(\boldsymbol{z}) = \operatorname{argmax}_{\boldsymbol{\theta} \in S_n(\boldsymbol{z})} l_n(\boldsymbol{\theta}, \boldsymbol{z}, 0)$.

4.2.2 Theoretical Results

In the following, we state our main theorem, which extends the safety and optimality guarantees from Sukhija et al. [1] to the contextual case.

We say that the solution to Equation (2), $\boldsymbol{\theta}^*(\boldsymbol{z})$, is *discoverable* if there exists a finite $\tilde{n}$ such that $\boldsymbol{\theta}^*(\boldsymbol{z}) \in \bar{R}_\epsilon^{\boldsymbol{z}}(S_{\tilde{n}}(\boldsymbol{z}))$. Here, $\bar{R}_\epsilon^{\boldsymbol{z}}(S) \subseteq \Theta$ represents the largest safe set which can be reached safely from $S \subseteq \Theta$ up to ϵ-precision (c.f., Equation (19) in Appendix A).

Theorem 1. *Consider any $\epsilon > 0$ and $\delta \in (0, 1)$. Further, let Assumptions 1 to 5 hold and $\beta_n(\delta)$ be defined as in Lemma 1. For any context $\boldsymbol{z} \in \mathcal{Z}$, let $\tilde{n}(\boldsymbol{z})$ be the smallest integer such that*

$$\frac{n(\boldsymbol{z})}{\beta_{\tilde{n}(\boldsymbol{z})}(\delta) \cdot \gamma_{n(\boldsymbol{z})|\mathcal{I}|}(\boldsymbol{z})} \geq \frac{C|\Theta|^2}{\epsilon^2} \qquad \textit{where} \quad n(\boldsymbol{z}) = \sum_{n=1}^{\tilde{n}(\boldsymbol{z})} \mathbb{1}\{\boldsymbol{z} = \boldsymbol{z}_n\} \tag{12}$$

and $C = 32/\log(1 + \sigma^{-2})$. Here, $\gamma_n(\boldsymbol{z}) = \max_{A \subset \Theta \times \mathcal{I}, |A| \leq n} \mathrm{I}(\boldsymbol{y}_{A,\boldsymbol{z}}; \boldsymbol{h}_{A,\boldsymbol{z}}) \leq \gamma_n$ denotes the mutual information between $\boldsymbol{h}_{A,\boldsymbol{z}} = [h(\boldsymbol{\theta}, \boldsymbol{z}, i)]_{(\boldsymbol{\theta},i) \in A}$ and corresponding observations.

Then, when running contextual GoSafeOpt *and if $\boldsymbol{\theta}^*(\boldsymbol{z})$ is discoverable, the following inequalities jointly hold with probability at least $1 - 2\delta$:*

1. $\forall t \geq 0, i \in \mathcal{I}_q \colon \bar{q}_i(\boldsymbol{x}(t), \boldsymbol{z}) \geq 0,$ *(safety)*
2. $\forall \boldsymbol{z} \in \mathcal{Z}, n \geq \tilde{n}(\boldsymbol{z}) \colon g(\hat{\boldsymbol{\theta}}_n(\boldsymbol{z}), \boldsymbol{z}) \geq g(\boldsymbol{\theta}^*(\boldsymbol{z}), \boldsymbol{z}) - \epsilon.$ *(optimality)*

It is natural to start for each $i \in \mathcal{I}$ with kernels $k_i^{\mathcal{Z}}$ and k_i^{Θ} on the space of contexts and the space of parameters, respectively, and to construct composite kernels $k_i = k_i^{\mathcal{Z}} \otimes k_i^{\Theta}$ or $k_i = k_i^{\mathcal{Z}} \oplus k_i^{\Theta}$ as the product or sum of the pairs of kernels (see section 5.1 of [3]). In this case, the information gain γ_n is sublinear in n for common choices of kernels $k_i^{\mathcal{Z}}$ and k_i^{Θ} implying that $n^*(\boldsymbol{z})$ is finite.

The theorem is proven in Appendix A.3. Comparing to contextual SafeOpt [17] which is only guaranteed to converge to safe optima in $\bar{R}_\epsilon^{\boldsymbol{z}}(S_0(\boldsymbol{z}))$, the global exploration steps of contextual GoSafeOpt can also "discover" a safe optimum which was not reachable from the initial safe seed.

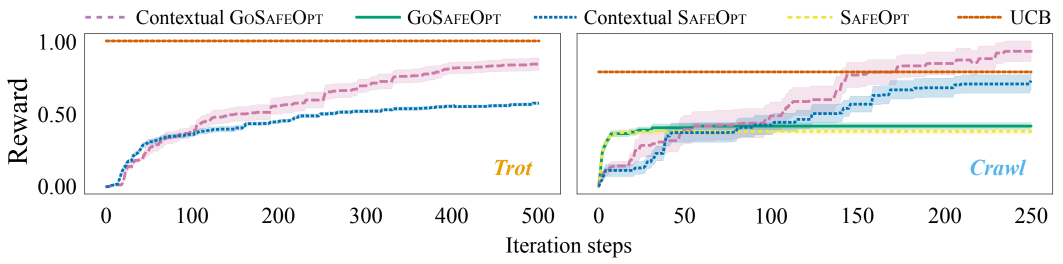

Figure 2: Simulation experiments. Left: Single context reward. GoSafeOpt outperforms SafeOpt on this task. On the right, we compare the learning curves of the contextual variants of GoSafeOpt and SafeOpt to the non-contextual ones for the *crawl* gait.

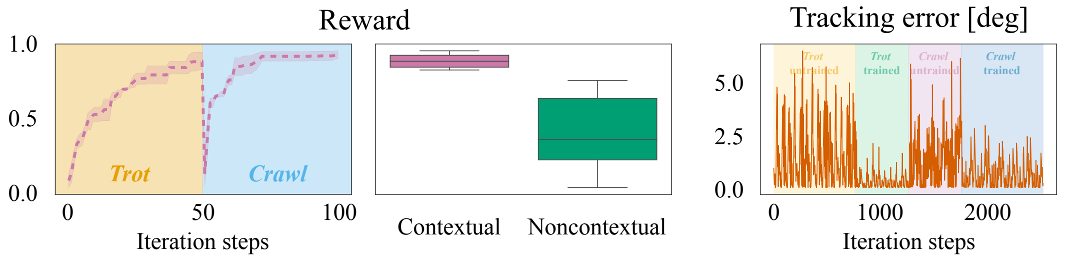

Figure 3: Hardware experiments. On the left, we present the learning curve of Contextual GoSafeOpt. It shows that the algorithm successfully tunes the controller gains for *trot*, and then subsequently for *crawl*. In the middle, the performance of the optimized control gains of *trot* on *crawl* to the optimized gains for *crawl* is compared. On the right, we present the tracking error of the hip joint for the front-left leg with *trot* and *crawl* gait at initialization (trot: yellow, crawl: violet) and after optimization (trot: green, crawl: blue). We see that the optimized gaits give a drastic reduction in tracking error.

We remark that Theorem 1 is a worst-case result and, in particular, disregards a possible statistical dependence between different contexts. In practice, if a kernel is chosen which does not treat all contexts as independent, then the convergence can be much faster as knowledge about a particular context can be transferred to other contexts.

5 Experimental results

We evaluate contextual GoSafeOpt on the *Unitree Go1* robot in both simulation and hardware. We provide a implementation, as well as a video of the simulation and hardware demonstrations in the supplementary. For both simulation and hardware, we use the following reward and constraints

$$g(\boldsymbol{\theta}, \boldsymbol{z}) = -\sum_{t\geq 0} \|\boldsymbol{x}^*(t) - \boldsymbol{x}(t, \boldsymbol{z}, \boldsymbol{\theta})\|^2_{\boldsymbol{Q}_g}, \quad q_0(\boldsymbol{\theta}, \boldsymbol{z}) = \min_{t\geq 0} v_0 - \|\boldsymbol{x}^*(t) - \boldsymbol{x}(t, \boldsymbol{z}, \boldsymbol{\theta})\|^2_{\boldsymbol{Q}_q}, \quad (13)$$

where $\boldsymbol{Q}_g$, and $\boldsymbol{Q}_q$ are positive semi-definite matrices. More details on the reward are presented in Appendix D. As a feedback policy, we use the PD controller introduced in Section 4.1.

Simulation experiments In simulation, we compare contextual GoSafeOpt to SafeOpt, and GoSafeOpt without contexts. Furthermore, we also evaluate GP-UCB [24], an unconstrained BO algorithm. To induce the 'sim-to-real-gap', we modify the simulation dynamics by adding additional disturbances to the system in the form of joint impedances at each joint (see Appendix D). The disturbances we induce can destabilize the system and thus cause constraint violation. Accordingly, we start all our experiments with a safe initial yet suboptimal feedback controller.

We optimize the controller for two different gaits; *trot*, and *crawl* sequentially. The parameters for the feedback controller and the parameterization of the gaits result in an overall model dimensionality of thirteen. We run all simulation experiments for ten different seeds and report the mean with one standard error. During all our experiments, we observe that all of the safe algorithms do not violate any safety constraint, while the standard GP-UCB method results in average constraint violation for 4.7% and 8% of all evaluations for *trot* and *crawl* gait, respectively. In Figure 2, we compare the normalized performance of GoSafeOpt and SafeOpt w.r.t. our objective for *trot* gait. It is visible from the figure that GoSafeOpt's global exploration helps in finding better controller parameters faster. Moreover, the GoSafeOpt performs nearly as well as GP-UCB, while violating no constraints. We also compare the contextual variants of GoSafeOpt and SafeOpt to the non-contextual ones. From the figure, we conclude that contextual variants find better optima, with contextual GoSafeOpt finding the best one. We believe this is because the contextual variants leverage the information collected while optimizing for *trot* gait, to find better optima for the crawl gait, and also avoid unsafe/unstable evaluations unlike GP-UCB.

Hardware Experiments We also evaluate contextual GoSafeOpt on the *Unitree Go1* robot. The robot has twelve motors in total. Even though we have a good model of the robot dynamics, the motors are typically difficult to model sufficiently accurately. Accordingly, we compensate for model imprecisions using controller gains. To this end, for motors in the same positions of each leg (e.g., motors on the hip joint) we use the same gains. In total, we have a six-dimensional parameter space.

Similar to the simulation experiments, we first tune the controller for *trot* gait and then for the crawl gait. We run the experiment over three different seeds and report the mean performance with one standard error. In all our experiments, GoSafeOpt results in zero constraint violations. Figure 3 show that GoSafeOpt is able to safely fine-tune the feedback control parameters of both contexts.

Since each gait has different dynamic properties, the best solution for one gait may not generalize well to other gaits. In Figure 3, in the middle, we depict how the best-performing parameters of *trot* gait do not generalize to the crawl gait, and learning the crawl context increases the reward by a considerable amount. From this, we conclude, that different gait patterns necessitate different controller gains.

Finally, in Figure 3, we also report the tracking performance of the tuned controller on the right. Moreover, we compare for both trot and crawl gaits the tracking performance of the initial and the tuned controller for the hip joint. We clearly see in the figure that the tuned controller makes considerably less tracking error. We provide the error plots for the remaining joints in Appendix D.

6 Conclusion

We have extended GoSafeOpt to incorporate the contextual setting and analyzed its convergence and safety properties theoretically. We showcased the efficacy of our algorithm through its application in adjusting the control parameters of a quadrupedal locomotion controller, in both simulation and hardware experiments. The use of contextual information enhances the convergence for new gait patterns by leveraging data from previously learned gaits. Furthermore, simulation results verify that GoSafeOpt can globally discover new safe regions without violating safety constraints. Across all of our experiments, GoSafeOpt outperforms prior approaches by a large margin and successfully finds optimal control parameters for different quadrupedal gait patterns. We highlight that the applicability of our proposed algorithm extends beyond our current scenario. It can be utilized to fine-tune any feedback controllers on actual hardware systems, making it an effective strategy to bridge the reality gap across various settings. For instance, we are interested in applying this method to a more diverse set of quadrupedal gait patterns and extend its scope to encompass non-periodic and unstructured gait patterns.

Limitations Even though the algorithm has theoretical safety guarantees, it is not always clear in practice if all theoretical assumptions are met. For instance, even though the surrogate model might be Lipschitz-continuous, the Lipschitz constant is generally not known a priori, i.e., Assumption 2 may not be satisfied. This often results in a too conservative choice of parameters, e.g., small lengthscale of the GP. Furthermore, a wrong parameter choice for the backup prior can result in unsafe global exploration or no global exploration at all. In general, safe exploration methods such as SafeOpt and GoSafeOpt have been successfully applied on several practical domains [1, 18, 19, 25, 26, 27], however closing the gap between theory and practice is still actively being researched [28, 29, 30].

References

[1] B. Sukhija, M. Turchetta, D. Lindner, A. Krause, S. Trimpe, and D. Baumann. Gosafeopt: Scalable safe exploration for global optimization of dynamical systems. *Artificial Intelligence*, 2023.

[2] Unitree Robotics. https://www.unitree.com/en/go1.

[3] A. Krause and C. Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, 2011.

[4] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems*, 2018.

[5] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 2019.

[6] D. Kang, J. Cheng, M. Zamora, F. Zargarbashi, and S. Coros. Rl + model-based control: Using on-demand optimal control to learn versatile legged locomotion, 2023.

[7] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control, 2019.

[8] D. Kang, F. De Vincenti, N. C. Adami, and S. Coros. Animal motions on legged robots using nonlinear model predictive control. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[9] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim. Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *IEEE Transactions on Robotics*, 2017.

[10] A. Schperberg, S. D. Cairano, and M. Menner. Auto-tuning of controller and online trajectory planner for legged robots. *IEEE Robotics and Automation Letters*, 2022.

[11] J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 1978.

[12] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth. Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker. *Annals of Mathematics and Artificial Intelligence*, 2016.

[13] M. Gelbart, J. Snoek, and R. Adams. Bayesian optimization with unknown constraints. *Conference on Uncertainty in Artificial Intelligence*, 2014.

[14] J. M. Hernández-Lobato, M. A. Gelbart, R. P. Adams, M. W. Hoffman, and Z. Ghahramani. A general framework for constrained Bayesian optimization using information-based search. *The Journal of Machine Learning Research*, 2016.

[15] A. Marco, D. Baumann, M. Khadiv, P. Hennig, L. Righetti, and S. Trimpe. Robot learning with crash constraints. *IEEE Robotics and Automation Letters*, 2021.

[16] Y. Sui, A. Gotovos, J. Burdick, and A. Krause. Safe exploration for optimization with Gaussian processes. In *International Conference on Machine Learning*, 2015.

[17] F. Berkenkamp, A. Krause, and A. P. Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *Machine Learning*, 2021.

[18] Y. Sui, V. Zhuang, J. Burdick, and Y. Yue. Stagewise safe Bayesian optimization with Gaussian processes. In *International Conference on Machine Learning*, 2018.

[19] C. König, M. Turchetta, J. Lygeros, A. Rupenyan, and A. Krause. Safe and efficient model-free adaptive control via bayesian optimization. In *IEEE International Conference on Robotics and Automation*, 2021.

[20] D. Baumann, A. Marco, M. Turchetta, and S. Trimpe. GoSafe: Globally optimal safe robot learning. In *IEEE International Conference on Robotics and Automation*, 2021. Proofs in extended online version: arXiv 2105.13281.

[21] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[22] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[23] S. R. Chowdhury and A. Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, 2017.

[24] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[25] A. Wischnewski, J. Betz, and B. Lohmann. A model-free algorithm to safely approach the handling limit of an autonomous racecar. In *IEEE International Conference on Connected Vehicles and Expo*, 2019.

[26] M. Fiducioso, S. Curi, B. Schumacher, M. Gwerder, and A. Krause. Safe contextual Bayesian optimization for sustainable room temperature PID control tuning. In *International Joint Conference on Artificial Intelligence*, 2019.

[27] S. E. Cooper and T. I. Netoff. Multidimensional bayesian estimation for deep brain stimulation using the safeopt algorithm. *medRxiv*, 2022.

[28] C. Fiedler, C. W. Scherer, and S. Trimpe. Practical and rigorous uncertainty bounds for Gaussian process regression. *AAAI Conference on Artificial Intelligence*, 35(8), 2021.

[29] F. Berkenkamp, A. P. Schoellig, and A. Krause. No-regret bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research*, 2019.

[30] J. Rothfuss, C. Koenig, A. Rupenyan, and A. Krause. Meta-learning priors for safe bayesian optimization. *arXiv preprint arXiv:2210.00762*, 2022.

[31] S. Vakili, K. Khezeli, and V. Picheny. On information gain and regret bounds in gaussian process bandits. In *International Conference on Artificial Intelligence and Statistics*, 2021.

[32] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[33] D. Kang, F. De Vincenti, and S. Coros. Nonlinear model predictive control for quadrupedal locomotion using second-order sensitivity analysis, 2022.

[34] K. Serkan, I. Turker, and G. Moncef. *Multidimensional particle swarm optimization for machine learning and pattern recognition*. Springer-Verlag Berlin Heidelberg, 2014.

[35] Constrained Bayesian optimization with particle swarms for safe adaptive controller tuning. *IFAC-PapersOnLine*, 2017.

[36] G. Pleiss, J. R. Gardner, K. Q. Weinberger, and A. G. Wilson. Constant-time predictive distributions for gaussian processes. *arXiv: 1803.06058*, abs/1803.06058, 2018.

Contents of Appendix

A Proofs

A.1 Definitions

We begin by re-stating definitions of sets used by GoSafeOpt from Sukhija et al. [1] with an additional context variable.

Fix an arbitrary context $\boldsymbol{z} \in \mathcal{Z}$. The *safe set* is defined recursively as

$$S_n(\boldsymbol{z}) = \bigcap_{i \in \mathcal{I}_q} \bigcup_{\boldsymbol{\theta}' \in S_{n-1}(\boldsymbol{z})} \{\boldsymbol{\theta} \in \Theta \mid l_n(\boldsymbol{\theta}', \boldsymbol{z}, i) - L_\Theta(\boldsymbol{z}) \|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \geq 0\} \tag{14}$$

where $L_\Theta(\boldsymbol{z})$ is the joint Lipschitz constant of g and the constraints q_i under context $\boldsymbol{z}$. The *expanders* are defined as

$$\mathcal{G}_n(\boldsymbol{z}) = \{\boldsymbol{\theta} \in S_n(\boldsymbol{z}) \mid e_n(\boldsymbol{\theta}, \boldsymbol{z}) > 0\} \qquad \text{with} \tag{15}$$

$$e_n(\boldsymbol{\theta}, \boldsymbol{z}) = |\{\boldsymbol{\theta}' \in \Theta \setminus S_n(\boldsymbol{z}) \mid \exists i \in \mathcal{I}_q \colon u_n(\boldsymbol{\theta}, \boldsymbol{z}, i) - L_\Theta(\boldsymbol{z}) \|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \geq 0\}| \tag{16}$$

and the *maximizers* are defined as

$$\mathcal{M}_n(\boldsymbol{z}) = \{\boldsymbol{\theta} \in S_n(\boldsymbol{z}) \mid u_n(\boldsymbol{\theta}, 0) \geq \max_{\boldsymbol{\theta}' \in S_n(\boldsymbol{z})} l_n(\boldsymbol{\theta}', 0)\}. \tag{17}$$

The analysis requires the ϵ-slacked safe region $\bar{R}_\epsilon^{\boldsymbol{z}}(S)$ given an initial safe seed $S \subseteq \Theta$, which is defined recursively as

$$R_\epsilon^{\boldsymbol{z}}(S) = S \cup \{\boldsymbol{\theta} \in \Theta \mid \exists \boldsymbol{\theta}' \in S \text{ such that } \forall i \in \mathcal{I}_q \colon q_i(\boldsymbol{\theta}', \boldsymbol{z}) - \epsilon - L_\Theta(\boldsymbol{z}) \|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \geq 0\}, \tag{18}$$

$$\bar{R}_\epsilon^{\boldsymbol{z}}(S) = \lim_{n \to \infty} (R_\epsilon^{\boldsymbol{z}})^n(S) \tag{19}$$

where $(R_\epsilon^{\boldsymbol{z}})^n$ denotes the nth composition of $R_\epsilon^{\boldsymbol{z}}$ with itself.

A.2 Algorithm

A.2.1 Local Safe Exploration

During LSE, we keep track for each context $\boldsymbol{z} \in \mathcal{Z}$ of a set of backup policies $\mathcal{B}(\boldsymbol{z}) \subseteq \Theta \times \mathcal{X}$ and observations of $\boldsymbol{h}$, which we denote by $\mathcal{D}(\boldsymbol{z}) \subseteq \Theta \times \mathbb{R}^{|\mathcal{I}|}$. An LSE step is described formally in Algorithm 1.

Algorithm 1 Local Safe Exploration (LSE)

```
Input: Current context $z_n$, safe sets $S$, sets of backups $\mathcal{B}$, datasets $\mathcal{D}$, Lipschitz constants $L_\Theta$
  1: Recommend parameter $\theta_n$ with Equation (10)
  2: Collect $\mathcal{R} = \bigcup_{k \in \mathbb{N}} \{(\theta_n, x(k))\}$ and $h(\theta_n, z_n, i) + \varepsilon_n$
  3: $\mathcal{B}(z_n) = \mathcal{B}(z_n) \cup \mathcal{R}$, $\mathcal{D}(z_n) = \mathcal{D}(z_n) \cup \{(\theta_n, h(\theta_n, z_n, i) + \varepsilon_n)\}$
  4: Update sets $S(z)$, $\mathcal{G}(z)$, and $\mathcal{M}(z)$ for all $z \in \mathcal{Z}$        ▷ Equations (14), (15) and (17)
Return: $S$, $\mathcal{B}$, $\mathcal{D}$
```

The LSE stage terminates for some given context $\boldsymbol{z} \in \mathcal{Z}$ when the connected safe set is fully explored and the optimum within the safe set is discovered. This happens when the uncertainty among the expanders and maximizers is less than ϵ and the safe set is not expanding

$$\max_{\boldsymbol{\theta} \in \mathcal{G}_{n-1}(\boldsymbol{z}) \cup \mathcal{M}_{n-1}(\boldsymbol{z})} \max_{i \in \mathcal{I}} w_{n-1}(\boldsymbol{\theta}, \boldsymbol{z}, i) < \epsilon \qquad \text{and} \qquad S_{n-1}(\boldsymbol{z}) = S_n(\boldsymbol{z}). \tag{20}$$

A.2.2 Global Exploration

A GE step conducts an experiment about a candidate parameter $\boldsymbol{\theta}_n \in \Theta$ which may not be safe. If the safety boundary is approached, GE conservatively triggers a safe backup policy. If, on the other hand, the experiment is successful, a new (potentially disconnected) safe region was discovered which can then be explored by LSE in the following steps. A GE step is described formally in Algorithm 2.

Algorithm 2 Global Exploration (GE)

```
Input: $z_n$, safe sets $S$, confidence intervals $C$, sets of backups $\mathcal{B}$, datasets $\mathcal{D}$, fail sets $\mathcal{E}$ and $\mathcal{X}_{\text{Fail}}$
 1: Recommend global parameter $\theta_n$ with Equation (11)
 2: $\theta = \theta_n$, $x_{\text{Fail}} = \emptyset$, Boundary = False
 3: while Experiment not finished do                                ▷ Rollout policy
 4:     if Not Boundary then
 5:         Boundary, $\theta_s^*$ = BoundaryCondition($z_n, x(k), \mathcal{B}$)
 6:         if Boundary then                                        ▷ Trigger backup policy
 7:             $\theta = \theta_s^*$, $x_{\text{Fail}} = x(k)$
 8:             $\mathcal{E} = \mathcal{E} \cup \{\theta_n\}$, $\mathcal{X}_{\text{Fail}} = \mathcal{X}_{\text{Fail}} \cup \{x_{\text{Fail}}\}$        ▷ Update fail sets
 9:     Execute until $x(k)$
10: Collect $\mathcal{R} = \bigcup_{k \in \mathbb{N}} \{(\theta_n, x(k))\}$, and $h(\theta_n, z_n, i) + \varepsilon_n$
11: if Not Boundary then                                            ▷ Successful global search
12:     $\mathcal{B}(z_n) = \mathcal{B}(z_n) \cup \mathcal{R}$ and $\mathcal{D}(z_n) = \mathcal{D}(z_n) \cup \{(\theta_n, h(\theta_n, z_n, i) + \varepsilon_n)\}$
13:     $S(z_n) = S(z_n) \cup \{\theta_n\}$
14:     $C(\theta_n, z_n, i) = C(\theta_n, z_n, i) \cap [0, \infty]$ for all $i \in \mathcal{I}_q$
Return: $S$, $C$, $\mathcal{B}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{X}_{\text{Fail}}$
```

A.2.3 Boundary Condition

The boundary condition checks when the system is in the state $\boldsymbol{x}$ whether there is a backup $(\boldsymbol{\theta}_s, \boldsymbol{x}_s) \in \mathcal{B}(\boldsymbol{z})$ such that $\boldsymbol{x}_s$ is sufficiently close to $\boldsymbol{x}$ to guarantee that $\boldsymbol{\theta}_s$ can steer the system back to safety for any state which may be reached in the next time step. If no such backups exist for the next states, a backup is triggered at the current state. In this case, the backup parameter $\boldsymbol{\theta}_s^*$ with the largest safety margin is triggered:

$$\boldsymbol{\theta}_s^* = \max_{(\boldsymbol{\theta}_s, \boldsymbol{x}_s) \in \mathcal{B}_n(\boldsymbol{z}_n)} \min_{i \in \mathcal{I}_q} l_n(\boldsymbol{\theta}_s, \boldsymbol{z}_n, i) - L_x \left\| \boldsymbol{x} - \boldsymbol{x}_s \right\|. \tag{21}$$

Algorithm 3 BoundaryCondition

```
Input: context $z_n$, state $x$, backups $\mathcal{B}$
 1: if $\forall(\theta_s, x_s) \in \mathcal{B}(z_n), \exists i \in \mathcal{I}_q : l_n(\theta_s, z_n, i) - L_x \|x - x_s\| + \Xi < 0$ then
 2:     Boundary = True, Calculate $\theta_s^*$ (Equation (21))
 3: else
 4:     Boundary = False, $\theta_s^* = \text{Null}$
return: Boundary, $\theta_s^*$
```

A.2.4 Contextual GoSafeOpt

The algorithm stops for a particular context $\boldsymbol{z} \in \mathcal{Z}$ when

$$\underbrace{\text{Equation (20) is satisfied}}_{\text{LSE converged}} \quad \text{and} \quad \underbrace{\Theta \setminus (S_n(\boldsymbol{z}_n) \cup \mathcal{E}(\boldsymbol{z}_n)) = \emptyset}_{\text{GE converged}}. \tag{22}$$

The full algorithm is described in Algorithm 4.

A.3 Proof of Theorem 1

Proof. We first derive the sample complexity bound of non-contextual GoSafeOpt. Then, we extend this sample complexity bound to contextual GoSafeOpt. We assume without loss of generality that β_n is monotonically increasing with n.

Sample complexity Assume first that the context is fixed, that is, $\forall n \geq 1 : \boldsymbol{z}_n = \boldsymbol{z}$. In this case, the safety guarantee (with probability at least $1 - \delta$) follows directly from Theorem 4.1 of Sukhija

Algorithm 4 Contextual GoSafeOpt

```
Input: Domain $\Theta$, Contexts $\mathcal{Z}$, Sequence of contexts $\{\boldsymbol{z}_n \in \mathcal{Z}\}_{n\geq 1}$, $k(\cdot,\cdot)$, $S_0$, $C_0$, $\mathcal{D}_0$, $\epsilon$
 1: Initialize GP $h(\boldsymbol{\theta}, \boldsymbol{z}, i)$, $\mathcal{E}(\boldsymbol{z}) = \emptyset$, $\mathcal{X}_{\text{Fail}}(\boldsymbol{z}) = \emptyset$, $\mathcal{B}_0(\boldsymbol{z}) = \{(\boldsymbol{\theta}, x_0) \mid \boldsymbol{\theta} \in S_0\}$
 2: while $\exists \boldsymbol{z} \in \mathcal{Z}$ such that GoSafeOpt has not terminated for $\boldsymbol{z}$ (Equation (22)) do
 3:     if GoSafeOpt has terminated for $\boldsymbol{z}_n$ (Equation (22)) then          ▷ Skip finished contexts
 4:         continue
 5:     for $\boldsymbol{x} \in \mathcal{X}_{\text{Fail}}(\boldsymbol{z}_n)$ do                                    ▷ Update fail sets
 6:         if Not BoundaryCondition($\boldsymbol{z}_n, \boldsymbol{x}, \mathcal{B}_n$) then
 7:             $\mathcal{E}(\boldsymbol{z}_n) = \mathcal{E}(\boldsymbol{z}_n) \setminus \{\boldsymbol{\theta}\}$, $\mathcal{X}_{\text{Fail}}(\boldsymbol{z}_n) = \mathcal{X}_{\text{Fail}}(\boldsymbol{z}_n) \setminus \{\boldsymbol{x}\}$
 8:     Update $C_n(\boldsymbol{\theta}, \boldsymbol{z}, i)$ $\forall \boldsymbol{\theta} \in \Theta$, $\boldsymbol{z} \in \mathcal{Z}$, $i \in \mathcal{I}$        ▷ Update confidence intervals, Equation (9)
 9:     if LSE not converged for context $\boldsymbol{z}_n$ (Equation (20)) then
10:         $S_{n+1}, \mathcal{B}_{n+1}, \mathcal{D}_{n+1} = \text{LSE}(\boldsymbol{z}_n, \mathcal{S}_n, \mathcal{B}_n, \mathcal{D}_n)$
11:     else
12:         $S_{n+1}, C_{n+1}, \mathcal{B}_{n+1}, \mathcal{D}_{n+1}, \mathcal{E}, \mathcal{X}_{\text{Fail}} = \text{GE}(\boldsymbol{z}_n, \mathcal{S}_n, C_n, \mathcal{B}_n, \mathcal{D}_n, \mathcal{E}, \mathcal{X}_{\text{Fail}})$
return: $\{\hat{\boldsymbol{\theta}}_n(\boldsymbol{z}) \mid \boldsymbol{z} \in \mathcal{Z}\}$
```

et al. [1]. Thus, it remains to show that the optimality guarantee with the given sample complexity holds also with probability at least $1-\delta$, as then their union holds jointly with probability at least $1-2\delta$ using a union bound.

It is straightforward to see (by employing Theorem 4.1 of Berkenkamp et al. [17]) that Theorem 4.2 of Sukhija et al. [1] holds for n^* being the smallest integer such that

$$n^* \geq \frac{C|\Theta|\beta_{n^*}(\delta)\gamma_{n^*|\mathcal{I}|}}{2\epsilon^2} \tag{23}$$

where we use that $|\bar{R}_0^{\boldsymbol{z}}(S)| \leq |\Theta|$ for any $S \subseteq \Theta$ and $|\Theta| + 1 \leq 2|\Theta|$. Thus, whenever a new disconnected safe region is discovered by GE, LSE is run for at most n^* steps.

It follows from the stopping criterion of GE, $\Theta \setminus (S_n \cup \mathcal{E}) = \emptyset$, that GE is run for at most $|\Theta|$ consecutive steps (i.e., without an LSE-step in between). Clearly, a new disconnected safe region can be discovered by GE at most $|\Theta|$ times, and hence, GoSafeOpt terminates after at most $|\Theta|$ iterations of at most n^* LSE steps and at most $|\Theta|$ GE steps. Altogether, we have that the optimality guarantee holds with probability at least $1-\delta$ for $\tilde{n}$ being the smallest integer such that

$$\tilde{n} = \left\lceil \frac{C|\Theta|^2\beta_{\tilde{n}}(\delta)\gamma_{\tilde{n}|\mathcal{I}|}}{\epsilon^2} \right\rceil \geq |\Theta|\left(n^* + |\Theta|\right), \tag{24}$$

completing the proof of Theorem 1 for non-contextual GoSafeOpt.

Multiple contexts Visiting other contexts $\mathcal{Z} \setminus \{\boldsymbol{z}\}$ in between results in additional measurements and increases the constant β, ensuring that the confidence intervals are well-calibrated. The only difference in the proofs is the appearance of $\beta_{n^*(\boldsymbol{z})}$ rather than $\beta_{n(\boldsymbol{z})}$ in Equation (23). In the contextual setting, $n^*(\boldsymbol{z})$ is the smallest integer such that

$$n(\boldsymbol{z}) \geq \frac{C|\Theta|\beta_{n^*(\boldsymbol{z})}(\delta)\gamma_{n(\boldsymbol{z})|\mathcal{I}|}(\boldsymbol{z})}{2\epsilon^2} \tag{25}$$

where

$$n(\boldsymbol{z}) = \sum_{n=1}^{n^*(\boldsymbol{z})} \mathbb{1}\{\boldsymbol{z} = \boldsymbol{z}_n\}$$

counts the number of episodes with context $\boldsymbol{z}$ until episode $n^*(\boldsymbol{z})$. The bound on $\tilde{n}(\boldsymbol{z})$ then follows analogously to Equation (24). □

Table 1: Here we summarize different magnitudes of γ_n for composite kernels from Theorems 2 and 3 of Krause and Ong [3] and for individual kernels from Theorem 5 of Srinivas et al. [24] and Remark 2 of Vakili et al. [31]. The magnitudes hold under the assumption that the domain of the kernel is compact. γ^{Θ} and $\gamma^{\mathcal{Z}}$ denote the information gain for the kernels k^{Θ} and $k^{\mathcal{Z}}$, respectively. B_ν is the modified Bessel function.

Kernel	$k(\boldsymbol{v}, \boldsymbol{v}')$	γ_n
Product	$k^{\Theta}(\boldsymbol{v}, \boldsymbol{v}') \cdot k^{\mathcal{Z}}(\boldsymbol{v}, \boldsymbol{v}')$ if $k^{\mathcal{Z}}$ has rank at most d	$d\gamma_n^{\Theta} + d\log(n)$
Sum	$k^{\Theta}(\boldsymbol{v}, \boldsymbol{v}') + k^{\mathcal{Z}}(\boldsymbol{v}, \boldsymbol{v}')$	$\gamma_n^{\Theta} + \gamma_n^{\mathcal{Z}} + 2\log(n)$
Linear	$\boldsymbol{v}^\top \boldsymbol{v}'$	$\mathcal{O}\left(d\log(n)\right)$
RBF	$e^{-\frac{\\|\boldsymbol{v}-\boldsymbol{v}'\\|^2}{2l^2}}$	$\mathcal{O}\left(\log^{d+1}(n)\right)$
Matérn	$\frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}\\|\boldsymbol{v}-\boldsymbol{v}'\\|}{l}\right)^{\nu} B_\nu\left(\frac{\sqrt{2\nu}\\|\boldsymbol{v}-\boldsymbol{v}'\\|}{l}\right)$	$\mathcal{O}\left(n^{\frac{d}{2\nu+d}}\log^{\frac{2\nu}{2\nu+d}}(n)\right)$

B Comparison of SafeOpt and GoSafeOpt

To visually analyze the different exploration properties of SafeOpt and GoSafeOpt we use the Pendulum Environment from OpenAI [32] as an example. The ideal trajectory is given by some undisturbed controller. In our toy problem, we use a simple PD control which is sufficient for the pendulum swing-up problem and various oscillating trajectories. To simulate the sim to hardware gap, we artificially add a disturbance to the applied torque in the form of joint impedances $\boldsymbol{\tau} = \boldsymbol{\tau}^* - \boldsymbol{\theta}_p^d(\boldsymbol{x}^* - \boldsymbol{x}) + \boldsymbol{\theta}_d^d\dot{\boldsymbol{x}}$ where $\boldsymbol{\theta}_p^d$ and $\boldsymbol{\theta}_d^d$ are unknown disturbance parameters and $\boldsymbol{x}^*, \boldsymbol{x}$ are the desired and observed motor angles. We use GoSafeOpt to tune an additional PD controller which should follow the ideal trajectory and compensate for the artificial disturbance. Figure 4 shows an example run of SafeOpt and GoSafeOpt. Whereas SafeOpt is restricted to expanding the initial safe region, GoSafeOpt can discover new safe regions, and thus find a better optimum.

C Control Formulation

Our model-based motion controller integrates the MPC and WBC methods to enhance both robustness and maneuverability. The MPC is responsible for generating base and foot trajectories, while the WBC converts these trajectories into joint-level commands. For the MPC component, we employ the model predictive control formulation proposed by Kang et al. [33, 8]. This formulation represents a finite-horizon optimal control problem as a nonlinear program utilizing the variable-height inverted pendulum model. The optimal solution of the nonlinear program is determined by using a second-order gradient-based method. For a more in-depth understanding of the MPC formulation, we direct readers to the prior work by Kang et al. [33, 8].

We employ a slight modification of the WBC formulation introduced by Kim et al. [7], adapting it to align with the MPC. Following the method proposed by Kim et al. [7], we compute the desired generalized coordinates $\boldsymbol{x}^{\text{cmd}}$, speed $\dot{\boldsymbol{x}}^{\text{cmd}}$, and acceleration $\ddot{\boldsymbol{x}}^{\text{cmd}}$ for a quadruped system at the kinematic level. This process involves translating desired task space (Cartesian space) positions, velocities, and accelerations into configuration space counterparts. Throughout this process, we enforce task priority through iterative null-space projection [21]. The top priority is assigned to the contact foot constraint task, followed by the base orientation tracking task. The base position tracking task is given the third priority, and the swing foot tracking task is assigned the final priority.

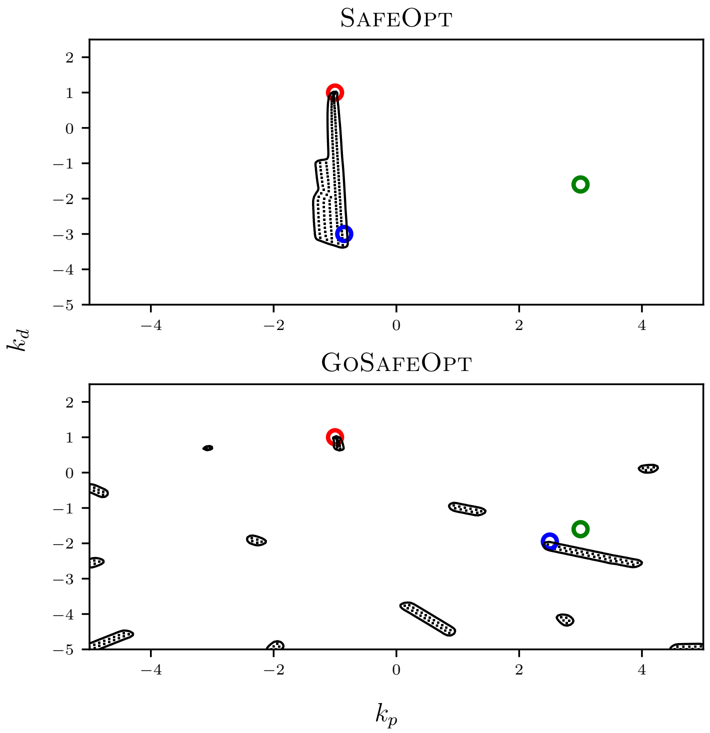

Figure 4: Example run of SAFEOPT and GOSAFEOPT. The red circle denotes the initial safe point. The black dots denote observed points. The green circle denotes the true safe optimum and the blue circle denotes the optimal point determined by SAFEOPT and GOSAFEOPT, respectively, after 150 iterations. The discovered safe sets are shown in black. GOSAFEOPT gets closer to the true optimum by discovering new safe regions which are not connected to the initial safe region.

Subsequently, we solve the following quadratic program:

$$\begin{aligned} \min_{\boldsymbol{\delta}_{\ddot{x}}, \boldsymbol{f}_c} \quad & \|\boldsymbol{\delta}_{\ddot{x}}\|^2_{\boldsymbol{Q}} && \text{(26a)} \\ \text{s.t.} \quad & \boldsymbol{S}_f(\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{b} + \boldsymbol{g}) = \boldsymbol{S}_f \boldsymbol{J}_c^\top \boldsymbol{f}_c && \text{(26b)} \\ & \ddot{\boldsymbol{x}} = \ddot{\boldsymbol{x}}^{\text{cmd}} + \left[\boldsymbol{\delta}_{\ddot{x}}, \boldsymbol{0}_{n_j}\right]^\top && \text{(26c)} \\ & \boldsymbol{W}\boldsymbol{f}_c \geq \boldsymbol{0}, && \text{(26d)} \end{aligned}$$

where $\boldsymbol{\delta}_{\ddot{x}}$ denotes a relaxation variables for the floating base acceleration and $\boldsymbol{f}_c$ denotes contact forces with the contact Jacobian $\boldsymbol{J}_c$. Equation (26a) is the objective function that penalizes the weighted norm of $\boldsymbol{\delta}_{\ddot{x}}$ with the weight matrix $\boldsymbol{Q}$. Equation (26b) corresponds to the equation of motion of the floating base, representing the first six rows of the whole-body equation of motion, with $\boldsymbol{S}_f$ being the corresponding selection matrix. Lastly, Equation (26d) sets forth the Coulomb friction constraints. This procedure refines the desired generalized acceleration $\ddot{x}^{\text{cmd}}$, which is calculated at the kinematic level, by incorporating the dynamic impacts of the robot's movements.

Upon determining $\ddot{x}$ is determined, we compute the joint torque commands as follows:

$$\boldsymbol{\tau} = \boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{b} + \boldsymbol{g} - \boldsymbol{J}_c^\top \boldsymbol{f}_c. \tag{27}$$

The final torque commands are calculated using $\boldsymbol{\tau}^{\text{cmd}} = \boldsymbol{\tau} + \boldsymbol{k}_p(\boldsymbol{x}^* - \boldsymbol{x}) + \boldsymbol{k}_d(\dot{\boldsymbol{x}}^* - \dot{\boldsymbol{x}})$ and dispatched to the robot with the feedback gains $\boldsymbol{k}_p \in \mathbb{R}^{d_u \times d_x}$ and $\boldsymbol{k}_d \in \mathbb{R}^{d_u \times d_x}$. As previously noted, this step is crucial in dealing with model mismatches, specifically, the differences in joint-level behavior stemming from actuator dynamics and joint friction.

D Experimental Details

D.1 Bayesian optimization

For all our experiments, we use a Matérn kernel with $\nu = 1.5$ for the underlying Gaussian Process. The lenghtscales are fixed during the whole optimization process and set to

Table 2: Kernel lenghtscales

	lenghtscales
Simulation	$[0.1, 0.05, 0.1, 0.05, 0.1, 0.05, 0.1, 0.05, 0.1, 0.1, 0.1, 0.1, 0.1]$
Hardware	[0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.5, 0.5, 0.5, 0.5, 0.5]

where the first n parameters correspond to the $(\boldsymbol{k}_p, \boldsymbol{k}_d)$ pairs and the last parameters to the context. For all experiments, we use $\beta = 16$ for the LCB on the constraints.

D.2 Simulation

Disturbance model The full body controller of the quadruped is artificially disturbed by changing the desired torque of each motor. More specifically, an artificial disturbance α_l is applied to each motor of the four legs l, resulting in an applied motor torque $\boldsymbol{\tau}_{\text{applied}}^{i,l} = \alpha_l \boldsymbol{\tau}_{\text{desired}}^{i,l} + \boldsymbol{\theta}_l[\boldsymbol{q}_{l,i}^* - \boldsymbol{q}_{l,i}, -\dot{\boldsymbol{q}}_{l,i}]^\top$ for motors i of leg l.

Reward function The state $\boldsymbol{x} \in \mathbb{R}^{24}$ of all 12 motors is described by each motor angle and angular velocity. We set the matrices from Equation (13) to $\boldsymbol{Q}_g = \boldsymbol{I}^{24\times 24}$ and $\boldsymbol{Q}_q^{i,j} = \mathbb{1}\{i = j \wedge i <= 12\}$.

D.3 Hardware

We slightly modify the reward function for the hardware experiment and include a penalty term on the joint velocities.

$$\hat{g}(\boldsymbol{\theta}, \boldsymbol{z}) = g(\boldsymbol{\theta}, \boldsymbol{z}) - \|\boldsymbol{x}(t)\|_{\boldsymbol{Q}_p}^2$$

where the velocity state errors in $\boldsymbol{Q}_g$ and $\boldsymbol{Q}_q$ in Equation 13 are set to zero, since the observed joint velocities are noisy finite difference approximations of the joint angles. Furthermore, we define $\boldsymbol{Q}_p$ to only include the noisy joint velocity observations. More specifically, we define $\boldsymbol{Q}_q^{i,j} = \boldsymbol{Q}_q^{i,j} = \mathbb{1}\{i = j \wedge i <= 12\}$ and $\boldsymbol{Q}_p^{i,j} = \frac{1}{2}\mathbb{1}\{i = j \wedge i > 12\}$ and $\boldsymbol{Q}_q, \boldsymbol{Q}_q, \boldsymbol{Q}_p \in \mathbb{R}^{24\times 24}$. Experimental results have shown, that adding a penalty term on the joint velocities acts as a regulator to prefer solutions where motor vibrations are low. This has shown to improve overall convergence and to visibly avoid solutions where motor vibrations are high.

The tracking performance is evaluated for the crawl gait before and after learning the optimal parameters for the motors 1-3 as shown in Figure 1. Figure 5 shows that the optimal feedback control parameters drastically reduce motor vibrations and increase the tracking performance.

E Practical modifications

E.1 Boundary conditions

We use the idea from Sukhija et al. [1] to reduce computational complexity by defining an interior and marginal set. Intuitively, the interior set contains all observed states for which the safety margin is high and the marginal set includes all states where the safety margin is greater than a certain threshold. More formally, Sukhija et al. [1] defines the interior and marginal set as :

$$\Omega_{I,n} = \{\boldsymbol{x}_s \in \mathcal{X} \mid (\boldsymbol{\theta}, \boldsymbol{x}_s) \in \mathcal{B}_n : \forall i \in \mathcal{I}_q, l_n(\boldsymbol{\theta}, i) \geq \eta_u\} \tag{28}$$

$$\Omega_{M,n} = \{\boldsymbol{x}_s \in \mathcal{X} \mid (\boldsymbol{\theta}, \boldsymbol{x}_s) \in \mathcal{B}_n : \forall i \in \mathcal{I}_q, \eta_l \leq l_n(\boldsymbol{\theta}, i) < \eta_u\} \tag{29}$$

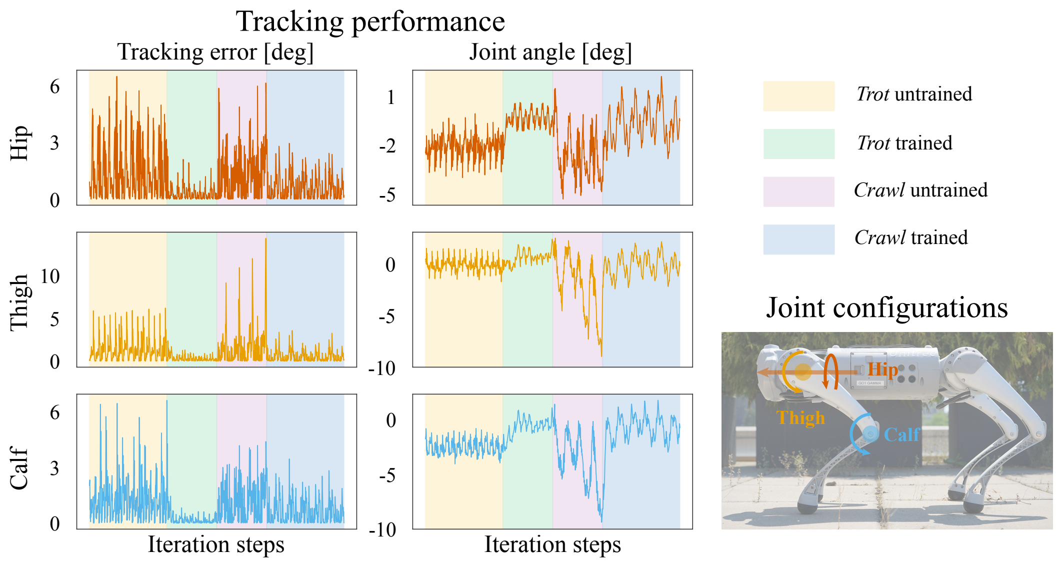

Figure 5: Joint angle tracking error in degrees (left) and joint angle measurement in degrees (right) for initialized controller gains for trot (yellow region) and crawl (violet region), and learned gains for trot (green) and crawl (blue). We clearly see that that the tracking error is considerably less for the tuned gaits and the motor commands also have less jitter.

The boundary condition is defined separately for the interior and marginal set. Firstly, the Euclidean distance d_i between the observed state and all the backup states is calculated. If $d_{\text{min}} = \min_i d_i = 0$, a backup policy for the observed state is known to be safe. Intuitively, the uncertainty if a backup policy can safely recover from the observed state increases as d_{min} grows. If the observed state moves too far away from the set of backup states, the closest backup policy is triggered. More formally, a backup policy is triggered, if the $\nexists d_i$ s.t $p(|x| \geq d_i) > \tau$. The distribution over x is defined as $x \sim \mathcal{N}(0, \sigma^2)$ and $\tau_m \geq \tau_i$ for the interior and marginal set, respectively. With σ^2 and τ_i there are two adjustable parameters to influence how conservative the backup policy acts.

Table 3: Boundary condition parameters

Parameter	Value	Description
Simulation		
σ	2	Standard deviation of backup distribution
τ_i	0.2	Interior lower bound probability
τ_m	0.6	Marginal lower bound probability
Hardware		
σ	2	Standard deviation of backup distribution
τ_i	0.05	Interior lower bound probability
τ_m	0.1	Marginal lower bound probability

E.2 Optimization

The solution of the acquisition optimization problem formulated in 11 is approximated with the standard particle swarm [34] algorithm, similar to [35].

At the beginning of each acquisition optimization, n_p particle positions are initialized. Rather than initializing the positions over the whole domain, the positions are sampled from a list of known safe positions in the current safe set.

For all experiments, the parameters in Table 4 are used.

Table 4: Swarmopt parameters

Parameter	Value	Description
Θ_g	1	Social coefficient
Θ_p	1	Cognitive coefficient
w	0.9	Inertial weight
n	100	Number of iterations
n_r	100	Number of restarts if no safe set is found

E.3 Fix iterations and discard unpromising new detected safe regions

In practice, it is not practical to fully explore a safe set before the global exploration phase. For our experiments, the number of iterations for the local and global exploration phase are fixed to $n_l = 10$ and $n_g = 5$, respectively. To avoid exploring for all n_l steps in unpromising regions, we define $n_d = 5 < n_l$ and switch to local exploration of the best set if the best reward estimation of the current set is much less than the best global reward estimate. That is, we switch to the best set if $\hat{r}_i^* < c\hat{r}^*$ and $n_d = n_l$.

E.4 Posterior estimation

Each BO step requires the optimization of the GoSafeOpt acquisition function to predict the next parameters to evaluate. This paper uses the standard particle swarm [34] algorithm, which requires the computation of the posterior distribution at each optimization step for all particles. To speed up the computation of the posterior distribution, the paper uses Lanczos Variance Estimates Pleiss et al. [36].