Sampling-Based System Identification with Active Exploration for Legged Robot Sim2Real Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Sim-to-real discrepancies hinder learning-based policies from achieving high-precision tasks in the real world. While Domain Randomization (DR) is commonly used to bridge this gap, it often relies on heuristics and can lead to overly conservative policies with degrading performance when not properly tuned. System Identification (Sys-ID) offers a targeted approach, but standard techniques rely on differentiable dynamics and/or direct torque measurement, assumptions that rarely hold for contact-rich legged systems. To this end, we present SPI-Active (Sampling-based Parameter Identification with Active Exploration), a two-stage framework that estimates physical parameters of legged robots to minimize the sim-to-real gap. SPI-Active robustly identifies key physical parameters through massive parallel sampling, minimizing state prediction errors between simulated and real-world trajectories. To further improve the informativeness of collected data, we introduce an active exploration strategy that maximizes the Fisher Information of the collected real-world trajectories via optimizing the input commands of an exploration policy. This targeted exploration leads to accurate identification and better generalization across diverse tasks. Experimental results demonstrate that SPI-Active enables precise sim-to-real transfer of learned policies to the real world, outperforming baselines by $42-63\%$ in various locomotion tasks. Videos at the anonymous website https://anonymous-spi-active.github.io/

Keywords: System Identification, Sim2Real, Legged Robots

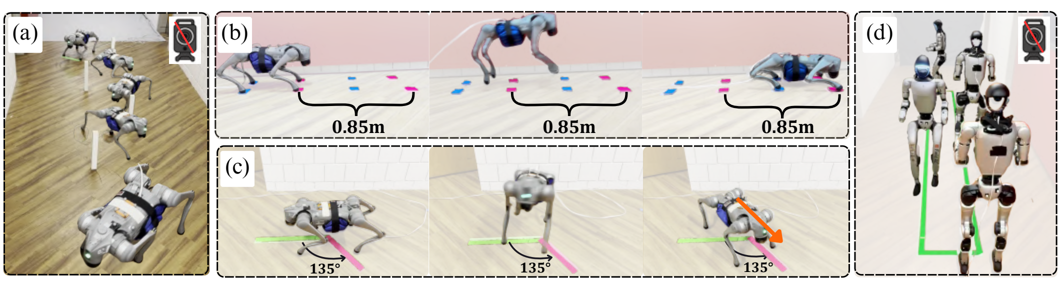


Figure 1: SPI-Active enables high-fidelity Sim-to-Real transfer across diverse locomotion tasks. To show the precision, all tasks are open-loop tracking *without global position feedback*. (a) High-Speed Weave Pole Navigation, (b) Precise Forward Jump, (c) Precise Yaw Jump, and (d) Humanoid Precise Velocity Tracking.

1 Introduction

Legged robots are envisioned to be used in complex environments where every stride demands a precision that leaves no room for error [1, 2]. Reinforcement Learning (RL) has shown remarkable success in enabling agile motions on both quadruped and humanoid systems [3, 4, 5, 6]. However, transferring RL policies from simulation to hardware remains challenging due to the sim-to-real gap. This gap primarily stems from mismatches in physical parameters such as mass, inertia, friction, and unmodeled effects in actuator dynamics and contact interactions, where even small discrepancies can severely degrade performance in the real world.

To bridge this gap, four broad strategies have been developed: **(1) Domain Randomization (DR)** trains robust policies by exposing them to wide parameter distributions in simulation [7, 8, 9, 10]; **(2) "White-Box" System Identification** directly estimates physical parameters using real-world data [11, 12, 13]; **(3) "Black-Box" System Identification** learns full or residual dynamics model [14, 15, 16] from ground-truth data; and **(4) Adaptive policy learning** adapts or fine-tunes policies online using real-world feedback [17, 18]. While practical, DR often requires heuristic tuning: excessive randomization leads to conservative policies, while insufficient randomization compromises real-world generalization. Approaches in (3) and (4) can be task-specific, prone to overfitting, and may demand substantial real-world data. In contrast, "White-Box" System Identification offers a principled, interpretable, and generalizable approach by estimating physically meaningful parameters, making it the focus of this work.

Despite its success in classic control, system identification for legged locomotion is challenging due to severe non-linearities and intermittent contacts. Many existing methods either assume differentiable [19] dynamics, rely on specialized sensing such as ground-truth torques [20], or estimate only a limited subset of parameters [21, 12], limiting their applicability to general-purpose legged systems. Another challenge is collecting sufficiently informative data for accurate estimation. Prior approaches often rely on hand-crafted motion scripts, simple repetitive behaviors, or isolated component tests [22, 23, 16]. While effective for subsystems, these fail to capture the coupled hybrid dynamics of natural locomotion and require task-specific tuning or extensive data collection.

In this work, we present SPI-Active, a two-stage, parallelizable, sampling-based framework for identifying structured physical parameters of legged robots—without requiring differentiable simulators or specialized sensing. In Stage 1, we leverage heuristically designed motion priors of pre-trained RL policies to collect real-world trajectories and estimate the robot physical parameters by minimizing state discrepancy between real and simulated rollouts. To enhance data efficiency and refine the initial estimates, Stage 2 draws on principles from optimal experiment design by maximizing the Fisher Information of the collected trajectories. Unlike prior work in manipulators [24], direct exploration policy training for legged robots can lead to erratic behaviors [25]. We address this by introducing a hierarchical active exploration strategy that optimizes command sequences of a multi-behavioral RL policy—targeting informative system excitation while ensuring reliable deployment. The refined parameters significantly improve sim-to-real transfer, enabling high-precision performance across diverse tasks on both the Unitree Go2 quadruped and the G1 humanoid, including challenging tasks (Figure. 1) such as precise jumping, high-speed weave pole traversal, and outperforming baselines by $42 - 63\%$. In summary, the main contributions are:

- A parallelized sampling-based system identification framework for legged robots that accounts for complex contact dynamics without specialized sensor requirements.
- An effective active exploration strategy that leverages command-space optimization of a multi-behavioural policy to induce highly informative data by maximizing Fisher Information.
- A comprehensive set of real-world experiments showcasing improvements in sim-to-real transfer and precise control in highly dynamic locomotion tasks.

2 Related Works

2.1 Domain Randomization and "Black-Box" System Identification for Sim2Real Transfer

A wide range of strategies have been developed to address the sim-to-real gap, including domain randomization, adaptive policy learning, and data-driven model learning.

Domain Randomization. Early efforts employed domain randomization (DR) to expose policies to diverse visual and physical variations during training [10], later extending to randomized dynamics [9, 22, 26, 27, 28] and sensor perturbations [29, 8]. While DR improves robustness, overly broad parameter ranges can lead to conservative policies that underperform on the true system. To address this, recent methods adapt the randomization process during training. Curriculum and adversarial DR strategies [30, 31] shape the parameter distributions over time, while others refine DR bounds

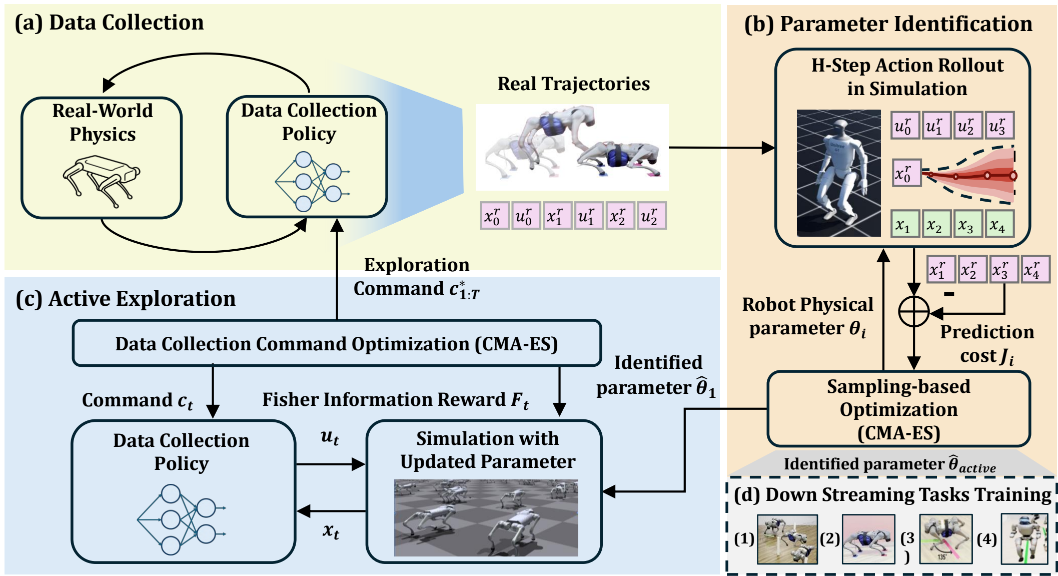


Figure 2: Overview of SPI-Active. **Data Collection:** Collect real-world trajectories using a multi-behavioral policy. **Parameter Identification:** Estimate physical parameters via simulation-to-real rollout matching by sampling-based optimization. **Active Exploration:** Optimize data-collection commands to maximize Fisher Information and gather informative data. **Downstream Task Training:** Use identified parameters to train accurate locomotion controllers

using real-world data [32]. Despite these advances, DR still relies heavily on heuristics, requiring expert tuning and task-specific knowledge.

Learning Adaptive Policies. Beyond DR, several approaches leverage online adaptation during deployment [33, 34, 17, 35] or use offline data for sim-to-real transfer [36] or condition the learnt policy on the prediction of model parameters online [37]. Some methods further adapt policies or simulation parameters during real-world rollouts to enable continual learning [18, 38], but these typically require high-quality data and are not zero-shot.

Data-driven Model Learning. Another class of methods learns data-driven residual networks that output corrective torques [39] or actions [40] to compensate for unmodeled dynamics. While effective in specific settings, these methods risk overfitting to the training tasks or trajectories, or requiring ground-truth torques [16], limiting their generalization to new or diverse tasks.

2.2 "White-Box" System Identification of Non-linear Dynamics

Modeling and identifying nonlinear dynamical systems remains challenging [41, 42, 19]. A foundational approach introduced least-squares estimation of inertial parameters via linearity in inverse dynamics [11], later refined with minimal parameter sets and model selection [43, 44, 45]. However, most methods assume structured and fixed-base models, limiting applicability to legged robots with discontinuous contacts and strong nonlinearities. Prior work mainly focuses on actuator modeling using analytical or hardware-specific approaches [22, 46, 23], while base parameter identification often requires constrained setups or physical disassembly [22, 47]. A related two-stage method [26] estimates actuator dynamics via latent-conditioned policies, but omits inertial parameters and lacks explicit torque decay modeling. In contrast, our framework jointly identifies inertial and actuator parameters with interpretable structure and improved sample efficiency.

2.3 Targeted Exploration for System Identification and Model Learning

Accurate System identification relies on collecting trajectories that sufficiently excite the dynamics of interest. Classic works on optimal experiment design [48, 49, 50] formalizes this using the Fisher Information Matrix (FIM) to reduce parameter uncertainty. Recent works extend this to nonlinear and hybrid systems: [25] optimize excitation for mechanical systems, while [51] leverage differentiable contact simulation to identify informative contact modes. Learning-based approaches such as ASID [24] and task-oriented exploration [52] actively excite dynamics via policy optimization. Others focus on scalable pipelines, including trajectory design benchmarks for inertial ID [53] and automated Real2Sim via robotic interaction [20]. Building on these foundations, we optimize command sequences of multi-behavioral policies to reliably excite informative dynamics, enabling scalable and robust parameter identification for high-dimensional legged robots.

3 SPI: Sampling-based Parameter Identification for Legged Robot

In this section, we introduce a zeroth-order system identification approach that leverages GPU-based parallel sampling to efficiently estimate the physical parameters of legged robots.

Preliminary: Consider the dynamics of the legged system given by: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t; \theta)$, where $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state, $\mathbf{u}_t \in \mathcal{U}$ is the control input, and $\theta \in \Theta \subseteq \mathbb{R}^d$ represents the unknown model parameters to be identified using state-action trajectories collected from the real-world system. Given a dataset $\mathcal{D}$ consisting of observed state-action sequences $\mathcal{D} = \{(\mathbf{x}_t, \mathbf{u}_t)\}_{t=1}^{N}$, the system identification problem can be formulated as the following optimization problem:

$$\theta^* = \arg\min_{\theta \in \Theta} \sum_{t=1}^{N} \|\mathbf{x}_{t+1} - f(\mathbf{x}_t, \mathbf{u}_t; \theta)\|^2. \tag{1}$$

where θ^* is the optimal robot parameter that minimizes the prediction error between the observed states from real-world data and predicted states from the model. Specifically for legged systems, we identify a set of physical parameters $\boldsymbol{\theta} = [\theta_{\text{in}}, \theta_{\text{mo}}]^T$, where θ_{in} represents the mass-inertia properties, including the mass, center of mass, and inertia of rigid bodies, and θ_{mo} denotes actuator model parameters that characterize motor dynamics. To ensure physical consistency in the mass-inertial parameterization, we apply the Log-Cholesky decomposition (Appendix A.4) to enforce it follows the Linear Matrix Inequality (LMI) constraint. In this work, we focus on identifying the robot's base link parameter identification. However, the approach can be readily extended to additional links with minimal modification.

Actuator Dynamics Modeling: In physics-based simulators, RL policies typically apply joint torques directly, with only torque limits enforced. However, real-world actuators exhibit significant non-linearities in high-torque regimes, leading to torque decay between commanded and actual outputs, which degrades performance in precision-critical or dynamic tasks. To better capture these effects, we formulate an actuator dynamics model, inspired from [54], using a hyperbolic tangent function to map desired to actual torques in simulation:

$$\tau_{\text{motor}} = \kappa \cdot \tanh\left(\frac{\tau_{\text{PD}}}{\kappa}\right), \qquad \tau_{\text{PD}} = \mathbf{K}_p(\mathbf{q}_{\text{target}} - \mathbf{q}) - \mathbf{K}_d\dot{\mathbf{q}}, \tag{2}$$

where $\mathbf{q}_{\text{target}}$, $\mathbf{q}$, $\dot{\mathbf{q}} \in \mathbb{R}^N$ denote target positions, joint states, and velocities. We assign joint-specific scaling parameters κ to accommodate differences in motor types and load conditions.

Data Collection and Pre-Processing: The goal of the data collection process is to induce diverse motion patterns improving parameter observability. To this end, we collect data for Stage 1 of SPI-Active using heuristically designed motion-priors and input command sequences of pre-trained RL locomotion policies (Figure. 2(a)). The resulting real-word trajectories form the dataset $\mathcal{D}_j = \{(\mathbf{x}_{t,j}, \mathbf{u}_{t,j})\}_{t=1}^{N_j}, \quad \mathcal{D} = \{\mathcal{D}_j\}$. To enhance the predictive capability of the system parameters, we extend the single-step system identification to a multi-step prediction formulation. Therefore, the dataset is segmented into various clips $\{c_k\}_{k=1}^{N_c}$ with horizon lengths H, following the simulation error criterion[13], where the length H is sampled from uniform distribution $\mathcal{U}(H_{\min}, H_{\max})$.Varying the clip length avoids bias introduced by fixed horizons and introduces an averaging effect that balances simulation error across trajectories. While this process, provides broad state-action coverage, the collected data might still lack targeted excitation of certain system parameters and designing heuristics for specific parameter excitation is often nontrivial. This limitation motivates the need for active trajectory design which we address in Section. 4.

Algorithm 1 SPI-ACTIVE: TWO-STAGE SAMPLING-BASED SYSID VIA ACTIVE EXPLORATION

Input: Data-collection policy $\pi(u_t \mid x_t, c_t)$, Simulator $f(x, u; \theta)$, Initial robot parameter θ_0
Output: Refined parameters $\hat{\theta}_{active}$

Stage 1: Initial Identification
$\mathcal{D}_0 \leftarrow$ collect real-world data using π with action primitives.
$\hat{\theta}_1 \leftarrow$ SPI $(\mathcal{D}_0, \theta_0)$
Stage2: Active Exploration and Refinement
$c^\star_{1:T} \leftarrow \arg\min_{c_{1:T}} \operatorname{tr}(\mathbf{F}(\hat{\theta}_1, \pi)^{-1})$ ▷ FIM optimization using CMA-ES
$\mathcal{D}_1 \leftarrow$ collect data using $\pi(u_t \mid x_t, c_t),\ c_t \sim c^\star_{1:T}$
$\hat{\theta}_{active} \leftarrow$ SPI $(\mathcal{D}_1, \hat{\theta}_1)$
return $\hat{\theta}_{active}$

SPI ($\mathcal{D}, \theta$):
Segment $\mathcal{D}$ into clips $\{c_k\}$, initialize CMA-ES with θ, Σ
repeat
    Sample $\{\theta_j\}_{j=1}^{B} \sim \mathcal{N}(\theta, \Sigma)$ ▷ Parallel rollouts
    for each θ_j **do**
        Evaluate trajectory prediction cost $J(\theta_j, \{c_k\})$ based on equation (3)
    Update CMA-ES using $J(\theta_j)$
until convergence
return $\arg\min_\theta J(\theta)$

Sampling-based Optimization Formulation: We formulate the identification problem as a non-linear least-squares H-Step Sequential prediction problem, with the cost function defined as:

$$J(\boldsymbol{\theta}, \{c_k\}) = \sum_{k=1}^{N} \sum_{t=0}^{H-1} \left\| \mathbf{x}^r_{t+1,k} - \mathbf{x}_{t+1,k} \right\|^2_{\mathbf{W}_x} + \|\theta - \theta_0\|^2_{\mathbf{W}_\theta}, \qquad \mathbf{x}_{t+1,k} = f(\mathbf{x}_{t,k}, \mathbf{u}^r_{t,k}; \boldsymbol{\theta}) \quad (3)$$

where f is the simulated dynamics conditioned on the parameter $\boldsymbol{\theta}$, $\mathbf{x}_{t,k}$ is the simulated state at timestep t and corresponding to clip c_k. The initial state for each clip is aligned with the real-world trajectory segment, and subsequent states are simulated using recorded control inputs $\mathbf{u}^r_{t,k}$. Given that main-stream RL-focused simulators such as Isaacgym are non-differentiable but parallelizable, we adopt a sampling-based optimization approach inspired by the recent works [55]. The optimal parameter estimate $\hat{\boldsymbol{\theta}}$ is found by minimizing $J(\theta)$ and this optimization is performed using the CMA-ES[56] framework within the optuna library [57]. At each iteration a batch of candidate parameter vectors $\{\boldsymbol{\theta}_j\}_{j=1}^{B}$ is sampled. The candidates are evaluated in parallel and $J(\boldsymbol{\theta_j})$ are used to update CMA-ES distribution until convergence (Figure. 2(b)).

4 SPI-Active: Active Exploration for Informative Data Collection

The performance of the Sampling-based System Identification framework Section. 3, depends heavily on the informativeness of the collected trajectories. Although Section. 3 uses heuristic design for data collection, this approach cannot fully excite the system dynamics. To improve data efficiency and enable more accurate parameter estimation, we introduce a principled trajectory excitation strategy that focuses on collecting a small set of highly informative trajectories. Cramer-Rao Bound [24] states that the covariance of any unbiased estimator $\hat{\theta}$ of the true parameters $\theta^\star$ is lower-bounded by the inverse of the FIM: $\mathbb{E}_{\mathbf{x}_{1:T} \sim p_{\theta^\star}} \left[(\hat{\theta} - \theta^\star)(\hat{\theta} - \theta^\star)^\top \right] \succeq \mathbf{F}(\theta^\star)^{-1}$, where the FIM of the parameterized trajectory distribution $p(\mathbf{x}_{1:T} \mid \boldsymbol{\theta}^\star)$ is given by:

$$\mathbf{F}(\boldsymbol{\theta}^\star) = \mathbb{E}_{\mathbf{x}_{1:T} \sim p(\cdot \mid \boldsymbol{\theta}^\star)} \left[\left(\frac{\partial}{\partial \boldsymbol{\theta}} \log p(\mathbf{x}_{1:T} \mid \boldsymbol{\theta}^\star) \right) \left(\frac{\partial}{\partial \boldsymbol{\theta}} \log p(\mathbf{x}_{1:T} \mid \boldsymbol{\theta}^\star) \right)^\top \right] \quad (4)$$

Intuitively, identifying an exploration policy π_{exp} that maximizes the FIM reduces the lower bound on the estimation variance. However, directly optimizing the FIM through an exploration policy may produce erratic behaviors. To this end, we propose a practical exploration strategy based on trajectory-level command optimization. Let $\pi(u_t|x_t, c_t)$ be a command-conditioned multi-behavioral policy/controller, where u_t is the control action, x_t is the system state and c_t is the command that modulates the velocities and locomotion behaviors. Rather than learning a exploration policy from scratch, we instead optimize over the command sequences $\mathbf{c_{1:N}}$, which enables the policy to generate diverse trajectories that can excite different modes of the underlying dynamics:

$$\mathbf{c}^{\star}_{1:T} = \arg\min_{\mathbf{c}_{1:T}} \operatorname{tr}(\mathbf{F}(\theta^{\star}, \pi)^{-1}) \tag{5}$$

Considering the dynamics Eq. 3 to have a Guassian process noise, $w_t \sim \mathcal{N}(0, \sigma^2 I)$, the $\mathbf{F}(\theta^{\star}, \pi)$ can be approximated with[24]:

$$\mathbf{F}(\theta^{\star}, \pi) \approx \sigma^{-2} \cdot \mathbb{E}_{p(\cdot|\hat{\theta}_1, \pi)} \left[\sum_{t=1}^{T} \frac{\partial f(x_t, u_t; \hat{\theta}_1)}{\partial \theta} \cdot \left(\frac{\partial f(x_t, u_t; \hat{\theta}_1)}{\partial \theta} \right)^{\top} \right] \tag{6}$$

This optimization is solved using the same CMA-ES optimizer described in Section. 3, to handle the non-differentiable dynamics and fully utilized the parallelization of the GPU-based simulator. Further, solving Eq. 4 requires θ^* which is not available in practice, hence we substitute it with the current best estimate $\hat{\theta}_1$ obtained from the Stage 1. Additional implementation details for the FIM maximization are provided in the Appendix A.3. The pseudo-code for the entire process in SPI-Active is provided in the Algorithm. 1.

5 Experiments

In this section, we present extensive experimentation results of our framework on both quadruped and humanoid systems. Through our experiments, we would like to answer the following questions:

1. Does SPI identify accurate robot models that match real-world dynamics?
2. Do the models identified by our methods enable improved sim2real transfer of RL policies for high-precision locomotion tasks?
3. Does the exploration strategy in SPI-Active further improve the performance of SPI?

5.1 Tasks and Hardware Overview

We evaluate the framework across the platforms, Unitree Go2 and Unitree G1. To examine the performance of sim2real transfer we consider four tasks namely: **Forward Jump, Yaw Jump, Velocity Tracking, Attitude Tracking** for the Unitree Go2 with an attached payload of 4.7 kg ($\sim$ one-third of its weight) and **Velocity Tracking** for Unitree G1. These tasks prominently exhibit the sim-to-real gap, and detailed task definitions and metrics can be found in the appendix A.1 and parameter identification results are in appendix A.6. For all the tasks, RL policies were trained with Isaac Gym [58] and we use Proximal Policy algorithm (PPO)[59] to maximize the cumulative discounted reward $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$.

Baselines: For each baseline, we specify the corresponding URDF and the Domain Randomization (DR) range for the trained RL policy, if applicable:

- *Vanilla*: Uses the nominal URDF with added payload and nominal DR range (Table 8, Column **1**).
- *Heavy DR*: Uses the same URDF as Vanilla but with a wider DR range (Table 8, Column **2**).
- *Gradient-based Sys-ID (**GD**)*: URDF parameters are identified using the gradient-based optimization method [60] with differentiable simulator (MJX [61]).
- *SPI*: Uses the URDF updated with parameters $\hat{\theta}_1$ from Stage 1 and nominal DR range.
- *SPI-Active*: Uses the URDF updated with parameters $\hat{\theta}_{active}$ from Stage 2 and nominal DR range.

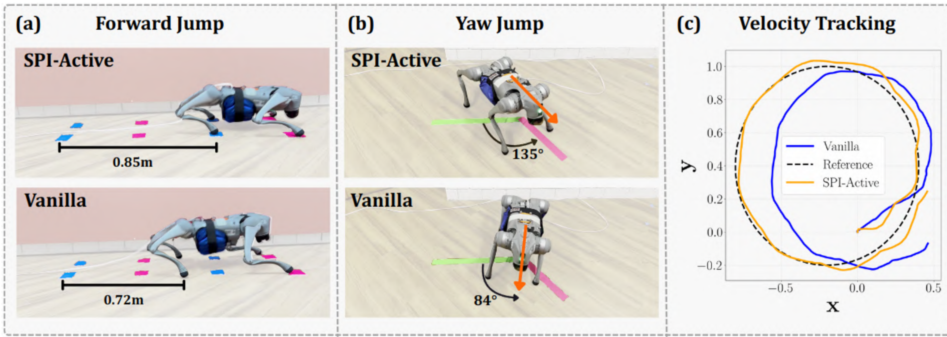


Figure 3: Task Performance comparison of SPI-Active vs Vanilla in (i) Forward Jump, (ii) Yaw Jump, (iii) and Velocity Tracking

Table 1: (a) Comparison of Prediction Accuracy and (b)Task Performance Across Methods

(a) Normalized Aggregate Prediction Error

Method	$J_{rpos}\downarrow$	$J_{pja}\downarrow$	$J_{rvel}\downarrow$
Vanilla	1.00	1.00	1.00
GD	0.98	1.14	1.05
SPI	0.87	0.89	0.95
SPI-Active	**0.67**	**0.72**	**0.91**

(b) Normalized Performance Across Tasks (↓)

Method	J_{fj}	J_{yj}	J_{vt}	J_{at}	J_{hvt}
Vanilla	1.00$\pm$0.000	1.00 $\pm$0.000	1.00 $\pm$0.000	1.00 $\pm$0.000	1.00 $\pm$0.066
Heavy DR	1.75$\pm$0.033	1.40 $\pm$0.021	1.06 $\pm$0.022	0.85$\pm$0.040	1.10 $\pm$0.064
SPI	0.60$\pm$0.014	0.64 $\pm$0.066	0.80 $\pm$0.044	0.96$\pm$0.042	0.87$\pm$0.064
SPI-Active	**0.48**$\pm$0.012	**0.37** $\pm$0.038	**0.58** $\pm$0.014	**0.73**$\pm$0.052	-

5.2 Open-Loop Prediction using Identified Model

To address **Q1**, we compare the prediction accuracy of the simulated trajectories with identified robot parameters of Unitree Go2 against real-world trajectories from a validation data. The data is collected by manually teleoperating Go2 for 60 seconds, while it runs the RL policy following[62]. We further preprocess the data, similar to Section.3 segmenting the data in various clips of average horizon length of 1.5sec. We evaluate the prediction accuracy by comparing the mean tracking error of the global root position J_{rpos}, per-joint angle J_{pja} and the global root velocity J_{rvel}. We report the normalized Quantitative results in Table. 1(a) demonstrates that SPI and SPI-Active consistently outperform the baselines, achieving lower J_{rpos} and J_{pja}, indicating more accurate open-loop prediction and closer alignment with real-world dynamics. In contrast, the gradient-based method ***GD*** exhibits larger prediction errors, possibly due to non-differentiable contact dynamics and the sim-to-sim gap between MJX and Isaac Gym.

5.3 Sim2Real Performance

To address **Q2**, we evaluate RL policies fine-tuned with system parameters identified by each method. Policies are trained for tasks shown in Appendix A.1 and deployed directly on hardware without further tuning. As reported in Table 1(b), both SPI and SPI-Active consistently outperform the baselines across all tasks. In *Velocity Tracking*, *Forward Jump* and *Yaw Jump*, SPI achieves an improvement of 19.6%, 39.9% and 35.9% respectively over the *Vanilla* baseline, highligting improved sim-to-real transfer (Figure. 3). While SPI under performs in the *Attitude Tracking* task - likely due to insufficient excitation of attitude-related system parameters, SPI-Active surpasses all baselines in *Attitude Tracking* (Appendix A.7) and across every task emphasizing the effectiveness of targeted excitation. Additionally, SPI improves performance in the *Humanoid Velocity Tracking* task, demonstrating the framework's generalization across robot morphologies and tasks.

5.4 Performance Improvement with Active Exploration

To investigate the effect of active exploration (**Q3**), we evaluate the impact of active exploration on real-world task performance by comparing three variants on the *Forward Jump* task: (1) SPI, (2) SPI +random where second-stage data is collected using randomly sampled input commands, and (3) SPI-Active. As shown in Fig. 4(c), SPI-Active yields a jumping distance error (3.6cm), lower compared to the other variants, demonstrating the effectiveness of targeted trajectory excitation for parameter identification. Further, Fig. 4(a) and (b) demonstrate that FIM-based command

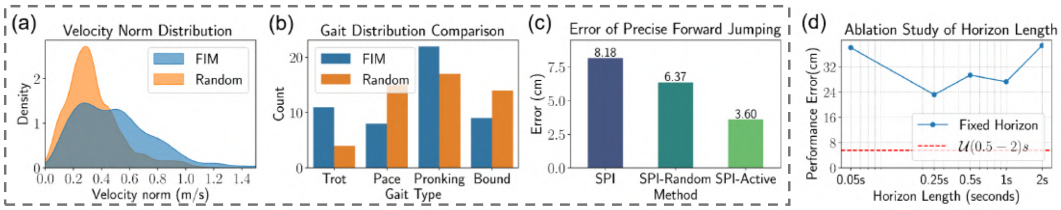


Figure 4: Effect of Active Exploration: (a) Velocity Magnitude Distribution (b) Gait Pattern Distribution and (c) Forward Jumping error comparison. Horizon Length Ablation (d)

optimization induces trajectories with higher velocity norm distributions and a higher occurrence of high-torque gaits such as pronking. This results in richer excitation of the system dynamics, which is critical for accurate parameter identification.

5.5 Ablation Studies

Role of Actuator Modeling in Policy Transfer: Here we discuss the performance impact of using different motor models. We evaluate four torque models: (1) **Vanilla** (ideal torques), (2) **Linear Gain**, (3) **Unified Tanh** with unified κ, and (4) **Ours** (motor-specific κ), where we choose unique κ_i for hip, calf and thigh motors of Unitree Go2. Real-world experiments were conducted on the *Forward Jump* and *Yaw Jump* tasks. We report the normalized task metrics with respect to the Vanilla baseline in Table 2. It can be observed that, our model achieves the lowest error in both tasks. In the *Forward Jump* task, our method and Unified Tanh outperform the vanilla baseline by 45% and 26% respectively, highlighting the benefit of modeling joint-specific nonlinearities for high-torque maneuvers. However, in the *Yaw Jump* task, the Unified Tanh underperforms the Vanilla baseline, which is likely due to the lower torque demands and the effect of angular inertia, where unified scaling fails to capture joint-specific behavior.

Table 2: Comparison of Motor Models on Normalized Task Metrics

Method	**Definition** ($\tau_{\text{motor}} \sim$)	$J_{fj} \downarrow$	$J_{yj} \downarrow$
Vanilla	τ_{PD}	1.00±0.00	1.00±0.00
Linear Gain	$\kappa \cdot \tau_{PD}$	1.30 ±0.023	1.54 ±0.011
Unified Tanh	$\kappa \cdot \tanh(\tau_{PD}/\kappa)$	0.74 ±0.009	1.17 ±0.019
Ours	$\kappa_\mathbf{i} \cdot \tanh(\tau_{PD}/\kappa_i)$	**0.55** ±0.015	**0.76**±0.042

Influence of Horizon Length on Policy Performance: We also investigate the effect of horizon length H when segmenting trajectory clips c_k during system identification, using the *Forward Jump* task. We compare fixed-length horizons (0.05s to 2s) against uniformly sampled horizons within this range. As shown in Figure 4(d), uniformly sampled horizons lead to better estimation and downstream performance. Results show that smaller H fails to capture long-term temporal dependencies needed for accurate estimation, while larger H suffers from instability of the open-loop action rollout. Uniformly sampling H during Stage 1 yields a better trade-off, enabling the identification process to benefit from both short and long-horizon dynamics.

6 Conclusion

We presented a two-stage, sampling-based system identification framework for legged robots that combines robust physical parameter estimation with an active trajectory excitation strategy to enable precise and scalable sim-to-real transfer. Our method does not rely on differentiable simulators or ground-truth torques, making it broadly applicable across different robotic platforms. By leveraging heuristic RL policies in the first stage and optimizing command sequences for Fisher Information in the second, our approach generates informative trajectories that excite key inertial and actuator parameters ensuring reliable hardware execution. Experimental results on the Unitree Go2 and G1 demonstrate significant improvements in tracking accuracy and task performance compared to domain randomization and baseline identification approaches. These results highlight the importance of accurate model identification and targeted data collection in bridging the sim-to-real gap for legged locomotion.

7 Limitations

While our framework demonstrates strong performance on quadrupeds, its application to humanoids is currently limited to Stage 1 identification. Extending active trajectory excitation to humanoid systems is a promising direction, but presents challenges due to their high dimensionality and safety-critical dynamics. Additionally, our method operates offline; enabling online or adaptive identification could improve real-time performance in dynamic settings. The approach also assumes access to a multi-behavioral policy for generating diverse motions, which may not generalize to novel morphologies. Lastly, while CMA-ES enables parallelizable optimization, it can be computationally demanding in high-dimensional spaces, motivating future work on more sample-efficient or uncertainty-aware alternatives.

References

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, Oct. 2020. ISSN 2470-9476. doi:10.1126/scirobotics.abc5986. URL https://www.science.org/doi/10.1126/scirobotics.abc5986.

[2] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62): eabk2822, Jan. 2022. ISSN 2470-9476. doi:10.1126/scirobotics.abk2822. URL https://www.science.org/doi/10.1126/scirobotics.abk2822.

[3] Y. Yang, G. Shi, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots. CAJun: Continuous Adaptive Jumping using a Learned Centroidal Controller, 2023. URL https://arxiv.org/abs/2306.09557.

[4] Z. Zhuang, S. Yao, and H. Zhao. Humanoid Parkour Learning, 2024. URL https://arxiv.org/abs/2406.10759.

[5] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi. Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion, 2024. URL https://arxiv.org/abs/2401.17583.

[6] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. ANYmal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, Mar. 2024. ISSN 2470-9476. doi:10.1126/scirobotics.adi7566. URL https://www.science.org/doi/10.1126/scirobotics.adi7566.

[7] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters. Robot Learning from Randomized Simulations: A Review, 2021. URL https://arxiv.org/abs/2111.00956.

[8] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Deep Drone Racing: From Simulation to Reality With Domain Randomization. *IEEE Transactions on Robotics*, 36(1):1–14, Feb. 2020. ISSN 1552-3098, 1941-0468. doi:10.1109/TRO.2019.2942989. URL https://ieeexplore.ieee.org/document/8877728/.

[9] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, Brisbane, QLD, May 2018. IEEE. ISBN 9781538630815. doi:10.1109/ICRA.2018.8460528. URL https://ieeexplore.ieee.org/document/8460528/.

[10] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 23–30, Vancouver, BC, Sept. 2017. IEEE. ISBN 9781538626825. doi:10.1109/IROS.2017.8202133. URL http://ieeexplore.ieee.org/document/8202133/.

[11] C. An, C. Atkeson, and J. Hollerbach. Estimation of inertial parameters of rigid body links of manipulators. In *1985 24th IEEE Conference on Decision and Control*, pages 990–995, Fort Lauderdale, FL, USA, Dec. 1985. IEEE. doi:10.1109/CDC.1985.268648. URL http://ieeexplore.ieee.org/document/4048448/.

[12] H. Mayeda, K. Yoshida, and K. Osuka. Base parameters of manipulator dynamic models. *IEEE Transactions on Robotics and Automation*, 6(3):312–321, June 1990. ISSN 2374-958X. doi:10.1109/70.56663. URL https://ieeexplore.ieee.org/document/56663/.

[13] T. Lee, J. Kwon, P. M. Wensing, and F. C. Park. Robot Model Identification and Learning: A Modern Perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 7(1): 311–334, July 2024. ISSN 2573-5144. doi:10.1146/annurev-control-061523-102310. URL https://www.annualreviews.org/content/journals/10.1146/annurev-control-061523-102310.

[14] M. P. Deisenroth and C. E. Rasmussen. Pilco: a model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 465–472, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[15] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9784–9790, 2019. doi:10.1109/ICRA.2019.8794351.

[16] 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*, 4(26): eaau5872, Jan. 2019. ISSN 2470-9476. doi:10.1126/scirobotics.aau5872. URL https://www.science.org/doi/10.1126/scirobotics.aau5872.

[17] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots, 2021. URL https://arxiv.org/abs/2107.04034.

[18] P. Wu, W. Xie, J. Cao, H. Lai, and W. Zhang. Loopsr: Looping sim-and-real for lifelong policy adaptation of legged robots, 2024. URL https://arxiv.org/abs/2409.17992.

[19] T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, Jan. 2011. ISSN 00051098. doi:10.1016/j.automatica.2010.10.013. URL https://linkinghub.elsevier.com/retrieve/pii/S0005109810004279.

[20] N. Pfaff, E. Fu, J. Binagia, P. Isola, and R. Tedrake. Scalable Real2Sim: Physics-Aware Asset Generation Via Robotic Pick-and-Place Setups, 2025. URL https://arxiv.org/abs/2503.00370.

[21] M. Gautier and W. Khalil. A direct determination of minimum inertial parameters of robots. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1682–1687, Philadelphia, PA, USA, 1988. IEEE Comput. Soc. Press. ISBN 9780818608520. doi:10.1109/ROBOT.1988.12308. URL http://ieeexplore.ieee.org/document/12308/.

[22] 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 *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018. ISBN 9780992374747.

doi:10.15607/RSS.2018.XIV.010. URL `http://www.roboticsproceedings.org/rss14/p10.pdf`.

[23] R. Grandia, E. Knoop, M. Hopkins, G. Wiedebach, J. Bishop, S. Pickles, D. Müller, and M. Bächer. Design and Control of a Bipedal Robotic Character. In *Robotics: Science and Systems XX*. Robotics: Science and Systems Foundation, July 2024. ISBN 9798990284807. doi:10.15607/RSS.2024.XX.103. URL `http://www.roboticsproceedings.org/rss20/p103.pdf`.

[24] M. Memmel, A. Wagenmaker, C. Zhu, D. Fox, and A. Gupta. ASID: Active Exploration for System Identification in Robotic Manipulation. Jan. 2024. URL `https://openreview.net/forum?id=pdhMe50hZI`.

[25] T. Lee, B. D. Lee, and F. C. Park. Optimal excitation trajectories for mechanical systems identification. *Automatica*, 131:109773, Sept. 2021. ISSN 00051098. doi:10.1016/j.automatica.2021.109773. URL `https://linkinghub.elsevier.com/retrieve/pii/S0005109821002934`.

[26] W. Yu, V. C. V. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3503–3510, 2019. URL `https://ieeexplore.ieee.org/document/8967890`.

[27] M. Mozifian, J. C. G. Higuera, D. Meger, and G. Dudek. Learning domain randomization distributions for training robust locomotion policies, 2019. URL `https://arxiv.org/abs/1906.00410`.

[28] J. Siekmann, Y. Godse, A. Fern, and J. Hurst. Sim-to-real learning of all common bipedal gaits via periodic reward composition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL `https://ieeexplore.ieee.org/document/9561248`.

[29] F. Sadeghi and S. Levine. CAD2RL: Real Single-Image Flight without a Single Real Image, 2016. URL `https://arxiv.org/abs/1611.04201`.

[30] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization, 2019. URL `https://arxiv.org/abs/1904.04762`.

[31] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[32] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979, May 2019. doi:10.1109/ICRA.2019.8793789. URL `https://ieeexplore.ieee.org/document/8793789/`. ISSN: 2577-087X.

[33] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik. Adapting Rapid Motor Adaptation for Bipedal Robots, 2022. URL `https://arxiv.org/abs/2205.15299`.

[34] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-Hand Object Rotation via Rapid Motor Adaptation, 2022. URL `https://arxiv.org/abs/2210.04887`.

[35] G. B. Margolis, X. Fu, Y. Ji, and P. Agrawal. Learning to See Physical Properties with Active Sensing Motor Policies, 2023. URL `https://arxiv.org/abs/2311.01405`.

[36] A. Bose, S. S. Du, and M. Fazel. Offline multi-task transfer rl with representational penalization, 2024. URL `https://arxiv.org/abs/2402.12570`.

[37] W. Yu, J. Tan, C. K. Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification, 2017. URL `https://arxiv.org/abs/1702.02453`.

[38] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world, 2021. URL https://arxiv.org/abs/2110.05457.

[39] N. Fey, G. B. Margolis, M. Peticco, and P. Agrawal. Bridging the sim-to-real gap for athletic loco-manipulation, 2025. URL https://arxiv.org/abs/2502.10894.

[40] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, Z. Yi, G. Qu, K. Kitani, J. Hodgins, L. J. Fan, Y. Zhu, C. Liu, and G. Shi. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills, 2025. URL https://arxiv.org/abs/2502.01143.

[41] L. Ljung. System Identification. In J. J. Benedetto, A. Procházka, J. Uhlíř, P. W. J. Rayner, and N. G. Kingsbury, editors, *Signal Analysis and Prediction*, pages 163–173. Birkhäuser Boston, Boston, MA, 1998. ISBN 9781461272731 9781461217688. doi:10.1007/978-1-4612-1768-8_11. URL http://link.springer.com/10.1007/978-1-4612-1768-8_11.

[42] H. Natke. System identification. *Automatica*, 28(5):1069–1071, Sept. 1992. ISSN 00051098. doi:10.1016/0005-1098(92)90167-E. URL https://linkinghub.elsevier.com/retrieve/pii/000510989290167E.

[43] M. Gautier, P.-O. Vandanjon, and A. Janot. Dynamic identification of a 6 dof robot without joint position data. *2011 IEEE International Conference on Robotics and Automation*, pages 234–239, 2011. URL https://api.semanticscholar.org/CorpusID:16755317.

[44] A. Janot, P.-O. Vandanjon, and M. Gautier. A generic instrumental variable approach for industrial robot identification. *IEEE Transactions on Control Systems Technology*, 22(1):132–145, 2014. doi:10.1109/TCST.2013.2246163.

[45] Y. Han, J. Wu, C. Liu, and Z. Xiong. An iterative approach for accurate dynamic model identification of industrial robots. *IEEE Transactions on Robotics*, 36(5):1577–1594, 2020. doi:10.1109/TRO.2020.2990368.

[46] S. Masuda and K. Takahashi. Sim-to-Real Transfer of Compliant Bipedal Locomotion on Torque Sensor-Less Gear-Driven Humanoid, 2022. URL https://arxiv.org/abs/2204.03897.

[47] B. Zhang, D. Haugk, and R. Vasudevan. System Identification For Constrained Robots, Aug. 2024. URL http://arxiv.org/abs/2408.08830. arXiv:2408.08830.

[48] M. Gevers, A. S. Bazanella, X. Bombois, and L. Miskovic. Identification and the Information Matrix: How to Get Just Sufficiently Rich? *IEEE Transactions on Automatic Control*, 54(12): 2828–2840, Dec. 2009. ISSN 0018-9286, 1558-2523. doi:10.1109/TAC.2009.2034199. URL http://ieeexplore.ieee.org/document/5325719/.

[49] X. Bombois, M. Gevers, R. Hildebrand, and G. Solari. Optimal experiment design for open and closed-loop system identification. *Communications in Information and Systems*, 11(3): 197–224, Mar. 2011. ISSN 2163-4548. doi:10.4310/CIS.2011.v11.n3.a1. URL https://link.intlpress.com/JDetail/1805791097244827649.

[50] L. Gerencser and H. Hjalmarsson. Adaptive input design in system identification. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 4988–4993, Seville, Spain, 2005. IEEE. ISBN 9780780395671. doi:10.1109/CDC.2005.1582952. URL http://ieeexplore.ieee.org/document/1582952/.

[51] H. Sathyanarayan and I. Abraham. Exciting Contact Modes in Differentiable Simulations for Robot Learning, 2024. URL https://arxiv.org/abs/2411.10935.

[52] J. Liang, S. Saxena, and O. Kroemer. Learning Active Task-Oriented Exploration Policies for Bridging the Sim-to-Real Gap, 2020. URL https://arxiv.org/abs/2006.01952.

[53] Q. Leboutet, J. Roux, A. Janot, J. R. Guadarrama-Olvera, and G. Cheng. Inertial Parameter Identification in Robotics: A Survey. *Applied Sciences*, 11(9):4303, May 2021. ISSN 2076-3417. doi:10.3390/app11094303. URL https://www.mdpi.com/2076-3417/11/9/4303.

[54] R. Grandia, E. Knoop, M. A. Hopkins, G. Wiedebach, J. Bishop, S. Pickles, D. Müller, and M. Bächer. Design and control of a bipedal robotic character. *arXiv preprint arXiv:2501.05204*, 2025.

[55] C. Pan, Z. Yi, G. Shi, and G. Qu. Model-based diffusion for trajectory optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=BJndYScO6o.

[56] N. Hansen. The CMA Evolution Strategy: A Tutorial, Mar. 2023. URL http://arxiv.org/abs/1604.00772. arXiv:1604.00772.

[57] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework, 2019. URL https://arxiv.org/abs/1907.10902.

[58] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning, Aug. 2021. URL http://arxiv.org/abs/2108.10470. arXiv:2108.10470.

[59] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms, Aug. 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347.

[60] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier. Differentiable simulation for physical system identification. *IEEE Robotics and Automation Letters*, 6(2):3413–3420, 2021. doi:10.1109/LRA.2021.3062323.

[61] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi:10.1109/IROS.2012.6386109.

[62] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. *Conference on Robot Learning*, 2022.

[63] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. URL https://arxiv.org/abs/1907.10902.

[64] C. D. Sousa and R. Cortesao. Inertia Tensor Properties in Robot Dynamics Identification: A Linear Matrix Inequality Approach. *IEEE/ASME Transactions on Mechatronics*, 24(1):406–411, Feb. 2019. ISSN 1083-4435, 1941-014X. doi:10.1109/TMECH.2019.2891177. URL https://ieeexplore.ieee.org/document/8603830/.

[65] C. Rucker and P. M. Wensing. Smooth parameterization of rigid-body inertia. *IEEE Robotics and Automation Letters*, 7(2):2771–2778, 2022. doi:10.1109/LRA.2022.3144517.

[66] C. L. Lab. Humanoidverse: A multi-simulator framework for humanoid robot sim-to-real learning. https://github.com/LeCAR-Lab/HumanoidVerse, 2025.

[67] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4), July 2018. ISSN 0730-0301. doi:10.1145/3197517.3201311. URL https://doi.org/10.1145/3197517.3201311.

A Appendix

A.1 Tasks Definition

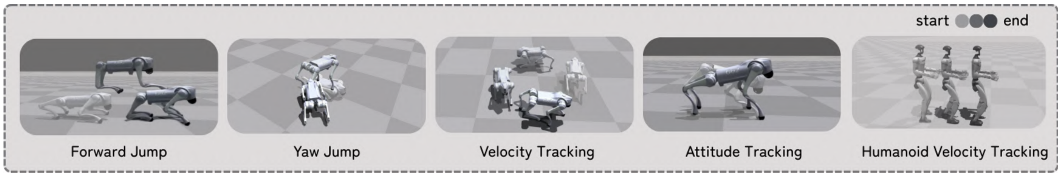


Figure 5: Open-Loop Locomotion Tasks: **Forward Jump**: Jump forward to a predefined distance of 0.85m. **Yaw Jump**: Jump and do Yaw Rotation to a predefined yaw angle of 135 degrees. **Velocity Tracking**: Track a sequence of Open loop 2D twist commands, **Attitude Tracking**: Track a sequence of roll and pitch commands, **Humanoid Velocity Tracking**: Track a sequence of 2D twist velocity commands for a humanoid.

A.1.1 Forward Jump

The forward jump task requires the robot to jump forward to a predefined horizontal distance x. For our experiments, we set $x = 0.85$ m. Additionally, during the training process, the policy was incentivized to maximize the vertical jump height, reaching a target height of $z = 0.35$ m above the initial height of the robot. To quantitatively evaluate the performance of the forward jump task, we define the performance error metric as:

$$J_{\text{fj}} = |x_f - x_i - 0.85| + |y_f - y_i| + |\max z - z_i - 0.35|,$$

where (x_i, y_i, z_i) and $(x_f, y_f, \max z)$ represent the robot's initial and final positions and the maximum height achieved during the jump, respectively. This metric captures the robot's ability to achieve the desired forward displacement while maintaining lateral stability ($y_f - y_i$) and achieving the target vertical jump height (0.35 m).

A.1.2 Yaw-Jump

The yaw jump task requires the robot to perform an in-place jump while achieving a specified yaw rotation. For our experiments, the target yaw angle was set to $\frac{3\pi}{4}$ radians. The performance error metric is defined as:

$$J_{\text{yj}} = |\phi_f| + |\theta_f| + |\psi_f - \psi_i - \frac{3\pi}{4}| + |x_f - x_i| + |y_f - y_i|,$$

where ϕ_f, θ_f, and ψ_f represent the final roll, pitch, and yaw angles in the robot's body frame, respectively, ψ_i is the initial yaw angle, and (x_i, y_i) and (x_f, y_f) denote the robot's initial and final horizontal positions. This metric accounts for the robot's ability to maintain stability in roll and pitch, achieve the desired yaw rotation of $3\pi/4$ radians, and minimize horizontal drift during the jump.

A.1.3 Velocity Tracking

In this task, the policy was trained to track velocity commands consisting of forward velocity (v_x), lateral velocity (v_y), and angular velocity (w_z). To evaluate the policy, we designed a circular trajectory tracking task, where the forward velocity command is varied linearly: it increases from $0.4\,\mathrm{m/s}$ to $0.8\,\mathrm{m/s}$ and then decreases back to $0.4\,\mathrm{m/s}$. The angular velocity command was adjusted accordingly to achieve a circular trajectory of radius $0.6\,\mathrm{m}$. The performance metric for this task is defined as:

$$J_{\text{vt}} = \sqrt{(v_{x,\text{ref}} - v_x)^2 + (v_{y,\text{ref}} - v_y)^2} + 0.5 \cdot |w_{z,\text{ref}} - w_z|, \tag{7}$$

where $v_{x,\text{ref}}$, $v_{y,\text{ref}}$, and $w_{z,\text{ref}}$ are the reference forward, lateral, and angular velocities, and v_x, v_y, and w_z are the actual tracked velocities. We follow the same definitions for the Humanoid Velocity Tracking task.

A.1.4 Attitude Tracking

In the attitude tracking task, the policy was trained to achieve commanded roll or pitch angles, all defined with respect to the body frame. During evaluation, the task involved tracking a periodic-ramp pitch reference signal with a fixed amplitude and frequency, followed by a periodic-ramp roll reference signal. The performance metric for this task is defined as:

$$J_{\text{rp}} = \sum_{i=1}^{N} \left\| \begin{bmatrix} \phi_i^{ref} - \phi_i \\ \psi_i^{ref} - \psi_i \end{bmatrix} \right\|_2 , \tag{8}$$

where ϕ_i^{ref} and ψ_i^{ref} are the sinusoidal reference signals for roll and pitch, respectively, and ϕ_i and ψ_i are the actual tracked roll and pitch angles over the evaluation period.

A.2 Implementation Details of SPI

The system identification objective is defined by the dynamics model in Eq. (1). The state vector $\mathbf{x}_t$ represents the full floating-base and joint state of the robot, defined as

$$\mathbf{x}_t = [\mathbf{p}_t, \mathbf{q}_t, \mathbf{v}_t, \boldsymbol{\omega}_t, \mathbf{q}_{\text{jnt},t}, \dot{\mathbf{q}}_{\text{jnt},t}]$$

where $\mathbf{p}_t \in \mathbb{R}^3$ is the base position, $\mathbf{q}_t \in \mathbb{R}^4$ is the base orientation represented as a unit quaternion, $\mathbf{v}_t \in \mathbb{R}^3$ is the linear velocity of the base, $\boldsymbol{\omega}_t \in \mathbb{R}^3$ is the angular velocity, $\mathbf{q}_{\text{jnt},t}$ denotes joint positions, and $\dot{\mathbf{q}}_{\text{jnt},t}$ denotes joint velocities.

The system identification cost function Eq.(3) in SPI consists of three components: **Base Prediction Cost**, which promotes global pose alignment by penalizing errors in the simulated floating-base state relative to motion-capture trajectories; **Joint Prediction Cost**, which enforces local dynamic consistency by minimizing discrepancies in joint position, velocity, and torque using proprioceptive data; and **Parameter Regularization**, which constrains deviations from nominal URDF values for physical and motor parameters, including mass, center of mass, inertia, and actuator gains.

The actuator model in Eq.(2) employs a hyperbolic-tangent form to approximate torque saturation. It preserves undisturbed torque output under small commands while smoothly saturating at the limits, ensuring both physical realism and optimization stability.

The full set of cost terms and their corresponding coefficients is detailed in Table 3. Coefficients are first normalized to yield unit cost on a reference dataset using default parameters, followed by global scaling: velocity-related terms are weighted by 0.5, torque-related terms by 0.2, and regularization terms by 0.1.

The parameter sampling ranges for CMA-ES initialization are detailed in Table 4, where each parameter is drawn from a uniform distribution centered at its nominal value specified in the URDF. Sampling-based optimization is performed using Optuna [63] with the default CMA-ES optimizer with Gaussian sampler, running for 5 iterations.

A.3 Implementation Details of SPI-Active

The active exploration in Stage-2 of SPI-Active requires us to optimize the input command sequences of a multi-behavioral policy(Section.4). To this end, we follow the training pipeline of [62] and pre-train an RL policy whose input commands c_t is a 14 dimensional vector, given by:

$$c_t = [v_x, v_y, w_z, h, f, b_1, b_2, b_3, b_4, h_f, \phi, \psi, s_w, s_l]^T \tag{9}$$

where v_x, v_y, w_z are the 2D velocity twist commands, and the policy is trained to track these commands, h, f, h_f refers the body height, stepping frequency and the foot swing height respectively. b_1, b_2, b_3, b_4 are the gait behavioral commands, that modify the quadrupedal gait and out of which b_4 determines the duration of the gait which is kept fixed even during the training phase. ϕ, ψ are the body roll and pitch commands, and s_w, s_l are the commanded stance width and length.

Table 3: SPI Cost Function Terms and Coefficients

Name	Expression	Coefficient
Base prediction cost		
Position prediction error	$\|p - p_r\|^2$	4.0
Velocity prediction error	$\|v - v_r\|^2$	2.0
Quaternion prediction error	$1.0 - \langle q, q_r \rangle^2$	2.0
Angular velocity prediction error	$\|\omega - \omega_r\|^2$	0.5
Joint prediction cost		
Joint position prediction error	$\|q_{jnt} - q_{jnt,r}\|^2$	3.0
Joint velocity prediction error	$\|\dot{q}_{jnt} - \dot{q}_{jnt,r}\|^2$	0.1
Joint torque prediction error	$\|\tau - \tau_r\|^2$	0.01
Parameter regularization		
Mass	$\|m - m_0\|^2$	0.01
Center of mass	$\|\mathbf{r} - \mathbf{r}_0\|^2$	10.0
Inertia	$\|\mathbf{I} - \mathbf{I}_0\|^2$	1.0
Tanh motor gain	$\|\kappa_{\tanh} - \kappa_{\tanh,0}\|^2$	0.01
Linear motor gain	$\|\kappa_s - \kappa_{s,0}\|^2$	0.1

Table 4: Sampling Ranges of Parameters for Different Robots

Name	Min	Max
Go2 (base link)		
Mass m	3.0	15.0
Inertia diagonal elements diag($\mathbf{I}$)	$(0.005, 0.005, 0.005)$	$(1.0, 1.0, 1.0)$
Center of mass $\mathbf{r}$	$(-0.1, -0.1, -0.1)$	$(0.1, 0.1, 0.1)$
Tanh motor gain $\kappa_{\tanh}$	$(10.0, 10.0, 10.0)$	$(40.0, 40.0, 40.0)$
Linear motor gain κ_s	0.5	1.5
G1 (pelvis link)		
Mass m	1.0	10.0
Inertia diagonal elements diag($\mathbf{I}$)	$(0.005, 0.005, 0.005)$	$(1.0, 1.0, 1.0)$
Center of mass $\mathbf{r}$	$(-0.2, -0.2, -0.2)$	$(0.2, 0.2, 0.2)$

In order to solve the optimization problem in Equation.5, we need to approximate the value of $\mathrm{tr}(\mathbf{F}(\theta^\star, \pi)^{-1})$ for a given trajectory. Hence, we consider our dynamics equation with a gaussian noise as given by:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t; \theta) + w_t \tag{10}$$

where $w_t \sim \mathcal{N}(0, \sigma^2 I)$, then the FIM reduces to:

$$\mathbf{F}(\theta^\star, \pi) = \sigma^{-2} \cdot \mathbb{E}_{p(\cdot|\theta^\star,\pi)} \left[\sum_{t=1}^{T} \frac{\partial f(x_t, u_t; \theta^\star)}{\partial \theta} \cdot \left(\frac{\partial f(x_t, u_t; \theta^\star)}{\partial \theta} \right)^\top \right] \tag{11}$$

Now, given that $\theta^\star$ is not available to us, we instead use a surrogate with $\hat{\theta}_1$ and given that our simulators need not be differentiable, we use finite difference approximation to calculate the gradient. Further, we add a termination penalty to prevent highly aggressive inputs commands that can lead to fall of the robot.

Further, If we want to collect a trajectory of length $\sim 40s \implies T = 2000$, to make this optimization problem more tractable, we constrain the input space by optimizing only a selected subset of commands: $[v_x, v_y, \omega_z, b_1, b_2, \phi, \psi]$ while keeping others fixed. These were chosen for their ability to sufficiently excite the physical parameters of interest during system identification. It is important to note that this subset is not fixed and can be adapted based on the specific parameters being targeted in different scenarios. Rather than optimizing these commands at every individual timestep, we adopt a more compact representation by reparameterizing the command trajectories using a 10-degree Bézier curve. This reduces the number of optimization variables, as we only sample and optimize the corresponding control points of the Bézier curve, excluding b_1 and b_2. The entire command sequence is divided into segments of fixed time horizons, each of length $H = 4$ seconds. For each horizon, the Bézier-defined command profile is resampled to generate the control sequence.For the gait-modulating commands b_1 and b_2, we select from four discrete combinations: $(0.5, 0.5)$, $(0.5, 0.0)$, $(0.0, 0.5)$, and $(0.0, 0.0)$, which correspond to pace, trot, bound, and pronk gaits respectively. These gait parameters are held fixed within each horizon to preserve consistent behavioral structure during execution.

A.4 Mass-Inertia Matrix Parameterization

The robot's inertial parameters θ_{in} includes: mass $m \in \mathbb{R}$, center of mass $\mathbf{r} \in \mathbb{R}^3$, and rotational inertia $\mathbf{I} \in S_3$ (the set of 3×3 symmetric matrices). These parameters are physcially feasible if and only if the pseudo-inertia $\mathbf{J}(\theta_{in})$ is positive definite [64]: $\mathbf{J}(\theta_{in}) = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{h} \\ \mathbf{h}^T & m \end{bmatrix} \succ 0$, where $\mathbf{h} = m \cdot \mathbf{r}$, $\mathbf{\Sigma} = \frac{1}{2}tr(\mathbf{I})\mathbb{I}_{3\times3} - \mathbf{I}$, and $\mathbf{I}_{3\times3}$ is the identity matrix. To ensure that these physical constraints are satisfied, we project the mass-inertia matrix $\mathbf{J}$ into a log-Cholesky form [65] that is feasible with respect to the positive-definite constraints.

A.5 Implementation details for Training Downstream Tasks

The training pipeline for all the downstream tasks uses the code framework inspired from [66]. We first mention the commanalities and then report the task specific rewards, observations and details.

For each task, we formulate it as a goal-conditioned Reinforcement learning (RL) task, where the policy π is trained to achieve a task and motivated to reduce the performance metrics in Appendix.A.1.Each policy is conditioned by observations o_t and it outputs action $a_t \in \mathbb{R}^{12}$ for the quadruped and $a_t \in \mathbb{R}^{15}$ for the humanoid, where the policy provides actions only to the lower body and the upper body joints are fixed. These actions correspond to the target joint positions and is passed to a PD controller that actuates the robot's degrees of freedom. The policy uses PPO [59] to maximize cumulative discounted reward. Further, we follow an asymmetric actor-critic training [40] to train the actor with only easily available observations from proprioception and other time based observations, while the critic has access to privileged information like base linear velocity which is usually difficult to estimate with on-board sensors.

A.5.1 Observations

All the policies use the robot's proprioception s_t^p and some task specific observations. The *Forward Jump* and *Yaw Jump* use a time phase variable Φ [67] to motivate the position of feet contacts to produce jump. The summary of observations are reported in Table5

Table 5: Observation Space for RL Policy

Component	**Description**
	Common Observations (used in all tasks)
$\boldsymbol{\omega}_t^{\text{base}}$	Base angular velocity
$\mathbf{g}_t$	Gravity vector projected in base frame
$\mathbf{q}_t$	Joint positions
$\dot{\mathbf{q}}_t$	Joint velocities
$\mathbf{a}_t$	Last applied actions
	Task-Specific Observations
Forward & Yaw Jump	Φ (phase), $\mathbf{a}_{t-1}$ (last-last action)
Velocity Tracking	v^x_{cmd}, v^y_{cmd}, ω^z_{cmd}
Attitude Tracking	ϕ_{cmd}(roll), ψ_{cmd}(pitch)
Humanoid Velocity Tracking	Φ (phase), $q^{\text{ref}_{\text{upper}}}$(upperbody dof reference)

A.5.2 Rewards

We summarize the weights of all reward terms used in tasks during policy training and evaluation. Quadruped task rewards are detailed in Table 6, covering different locomotion and agility objectives including block jumping, yaw jumping, tracking, and agile movement. Humanoid task rewards are shown in Table 7, with emphasis on gait stabilization, symmetry, and whole-body coordination.

Table 6: Reward terms for quadruped tasks

Term	Block Jump	Yaw Jump	Roll/Pitch Track	Agile Loco
Task Reward				
Body position	2.0	2.0	–	–
Body orientation	–	2.0	3.0	–
Body linear velocity	–	–	–	2.0
Body angular velocity	–	–	–	1.0
Feet height	3.0	3.0	–	–
Penalties & Regularization				
Action rate	–1e-3	–1e-3	–1e-3	–1e-2
Slippage	–3.0	–3.0	–1e-1	–
In-air contact	–3.0	–3.0	–	–
Foot spacing	–2.5	–2.5	–0.5	–
Non-foot contact	–	–0.3	–	–1e-1
Torque penalty	–	–	–2e-4	–2e-4
Acceleration penalty	–	–	–2.5e-7	–2.5e-7
Velocity penalty	–	–	–1e-4	–1e-4
Symmetry bonus	2.0	2.0	–	–
Base-height reference	–	–	1.0	2.0
Joint-limit violation	–	–	–10.0	–10.0

Table 7: Reward terms for humanoid velocity tracking

Term	Weight	Term	Weight
Task Reward			
Linear-velocity tracking	1.0	Angular-velocity tracking	1.0
Waist-joint tracking	0.5		
Penalties & Regularization			
Action-rate	–0.1	Vertical-vel	–2.0
Lateral-ang-vel	–0.05	Orientation	–1.5
Torque	–1e-5	Acceleration	–2.5e-7
Velocity	–1e-3	Contact-no-vel	–0.2
Feet-orientation	–2.0	Close-feet	–10.0
Joint-limit violation	–5.0	Base-height reference	–10.0
Contact	–0.20	Feet-heading alignment	–0.25
Hip-position	–1.0	Stance-tap	–5.0
Stance-root	–5.0	Stance-symmetry	–0.5
Survival (“alive”)	0.15	Contact bonus	0.18

A.5.3 Domain Randomization

We use two Domain Randomization ranges. The nominal range, that is used by the vanilla baseline, SPI and SPI-Active. However it should be noted that the motor parameter ranges $\kappa_{\text{Hip}}, \kappa_{\text{Thigh}}, \kappa_{\text{Calf}}$

corresponding to the Hip, Thigh and Calf are not used for the vanilla baseline. Second, we have the Heavy Range, where the ranges are almost double compared to the nominal range and is used by the *Heavy DR* baseline. The exact ranges are summarized in Table 8

Table 8: Domain Randomization Ranges

Term	Nominal Range	Heavy Range
	Dynamics Randomization	
Friction	$\mathcal{U}(0.5, 1.0)$	$\mathcal{U}(0.5, 1.0)$
Base CoM offset(m)	$\mathcal{U}(-0.1, 0.1)$	$\mathcal{U}(-0.2, 0.2)$
Base mass($\times$ default)Kgs	$\mathcal{U}(0.8, 1.2)$	$\mathcal{U}(0.6, 1.4)$
Base Inertia offset($Kg\mathrm{m}^2$)	$\mathcal{U}(-0.05, 0.05)$	$\mathcal{U}(-0.05, 0.3)$
κ_{Hip}	$\mathcal{U}(22, 24)$	-
κ_{Thigh}	$\mathcal{U}(24, 26)$	-
κ_{Calf}	$\mathcal{U}(22, 24)$	-
Term	Value	
Common Ranges		
P Gain	$\mathcal{U}(0.9, 1.1)$	
D Gain	$\mathcal{U}(0.9, 1.1)$	
Torque RFI	$0.1\times$ torque limit $\mathrm{N} \cdot \mathrm{m}$	

A.6 Parameter Identification Result Analysis

We evaluate the identified physical and actuator parameters of the Go2 robot with a 4.7kg payload mounted at the lower rear side of the base. The Table 9 compares the identified values against the default parameters without payload.

The identified model captures key physical changes introduced by the $4.7, \mathrm{kg}$ rear-mounted payload. The estimated mass increases by approximately $2.44, \mathrm{kg}$, partially compensating for the added load. The center of mass shifts rearward and downward, consistent with the payload's mounting location, and is essential for accurate contact force modeling and stability. Among the inertial parameters, we observe a reasonable increase in $\mathbf{I}_{zz}$, likely resulting from the added mass distribution around the yaw axis. However, the increases in $\mathbf{I}_{xx}$ and $\mathbf{I}_{yy}$ are unexpectedly large and not fully supported by the payload geometry. This suggests possible overfitting or parameter coupling due to insufficient excitation in the pitch and roll directions—an inherent challenge for quadruped systems. Nonetheless, the identified parameters yield improved trajectory prediction and real-world policy transfer performance, indicating that the model captures useful aspects of the true dynamics.

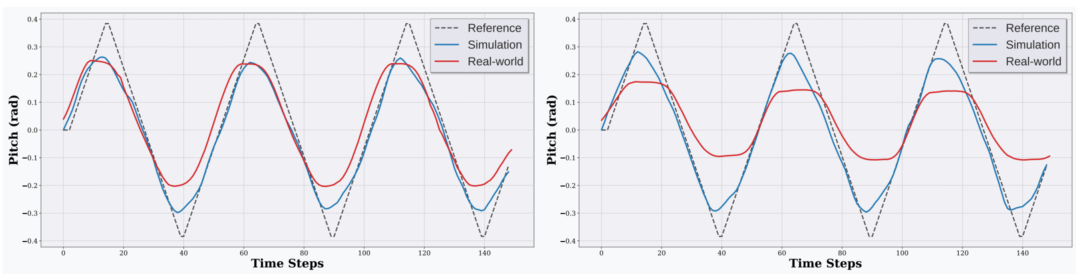
The actuator tanh gains reveal strong saturation effects at high-torque range. With gains around 25 for the thigh and calf joints, torque output saturates more gradually, resulting in a $20 - 26\%$ reduction near the maximum torque limits. Modeling this nonlinearity is critical for improving sim-to-real fidelity in high-torque tasks such as dynamic locomotion and jumping with payloads.

Table 9: Comparison of Default and Identified Parameters of Go2

Setting	**Mass**	**CoM**$_x$	**CoM**$_y$	**CoM**$_z$	$\mathbf{I}_{xx}$	$\mathbf{I}_{yy}$	$\mathbf{I}_{zz}$	κ_{Hip}	κ_{Thigh}	κ_{Calf}
Default	6.921	0.021	0.000	-0.005	0.025	0.098	0.107	—	—	—
Payload	9.363	0.004	-0.005	-0.020	0.391	0.515	0.396	22.553	24.969	23.523

A.7 Attitude Tracking Results

Figure 6 shows pitch tracking performance in an attitude control task, emphasizing the sim-to-real consistency of SPI-Active policies. The SPI-Active policy (left) exhibits a trajectory that closely matches the simulated response, whereas the vanilla policy (right) shows larger deviations from simulation. This demonstrates that SPI-Active achieves a smaller sim-to-real gap.



(a) SPI-Active

(b) Vanilla

Figure 6: Open-loop attitude tracking results. Comparison between SPI-Active and vanilla policies in both simulation and real-world execution. SPI-Active yields a closer match to simulation, suggesting improved sim-to-real consistency.