Safe and Efficient Quadrupedal Locomotion With a Chambolle–Pock Whole-Body Controller

Xu Yang, Run Wang, Yiwen Lu, and Yilin Mo, *Member, IEEE*

***Abstract*—This article presents a hierarchical control framework for quadrupedal locomotion that unifies the complementary strengths of model-based optimization and reinforcement learning. We develop a convex quadratic programming (QP) solver based on the primal-dual Chambolle–Pock algorithm, enabling both massively parallel policy training and real-time deployment through efficient handling of constrained optimization problems. Our hierarchical framework employs learned policies for robust high-level control to handle real-world perturbations, while ensuring instantaneous constraint satisfaction and energy efficiency through a low-level whole-body controller powered by the proposed solver. Extensive benchmarks and experimental validation demonstrate quantifiable improvements in energy consumption, constraint satisfaction, and task transferability across simulated and real-world environments.**

***Index Terms*—Legged locomotion, optimal control, reinforcement learning (RL), whole-body control.**

I. Introduction

QUADRUPED robots offer promising solutions for complex tasks in search and rescue operations, industrial inspections, and space exploration [1], [2], [3]. They are capable of traversing obstacles and maneuvering in environments inaccessible to wheeled or tracked vehicles, offering substantial benefits in scenarios requiring heavy load handling and agile movement. However, realizing these advantages requires addressing the critical concerns of safety, energy efficiency, and robustness.

The control of quadruped robots is inherently challenging due to their high-dimensional, underactuated, nonlinear, and hybrid dynamics with multiple physical and safety constraints [4]. Dominant control strategies for legged robots typically involve formulating an optimization problem to capture the desired behavior and safety constraints, which is then solved via one of two approaches: online numerical optimization or offline training through trial-and-error interactions. These methodologies represent model-based optimal control (OC) and reinforcement learning (RL), respectively. While recent advancements in both OC and RL have demonstrated significant progress in legged locomotion, each approach presents limitations that we examine in detail as follows.

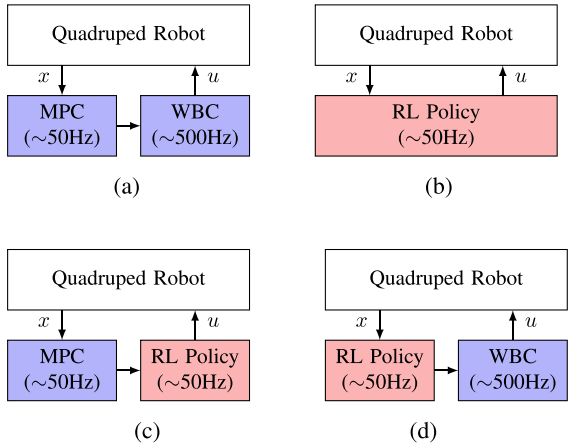

Fig. 1. Paradigms of quadrupedal locomotion control. We use u and x to denote the control input and state, respectively. We use blue for OC and red for RL. Typical operational frequencies are shown in parentheses. (a) MPC + WBC. (b) End-to-end RL. (c) OC + RL. (d) *RL + OC (Ours)*.

OC: In OC, the control law is computed by numerically solving an optimization problem online, where the objective function seeks to balance task performance with energy efficiency, while hard constraints are applied to enforce the robot's safety. However, the high dimensional, nonsmooth, and nonconvex nature of the optimization problem leads to high computational complexity, making it challenging to solve in real time. To mitigate the computational burden, a hierarchical control architecture, consisting of a low-frequency planner and a high-frequency tracking controller, is commonly employed [5], [6]. As illustrated in Fig. 1(a), this architecture separates control into a high-level model predictive controller (MPC) that generates reference trajectories using simplified dynamics models [7], [8], [9], [10], and a low-level whole-body controller (WBC) operating at higher frequencies to track these trajectories [11].

The high-level MPC typically uses reduced-order models, such as the spring-loaded inverted pendulum model [12] or the single rigid body dynamics (SRBD) model [13] to predict future states and optimize control inputs that minimize a given cost function. At the low level, WBC computes joint torques reactively based on the current state to track desired trajectories, with the optimization formulated as a QP problem that can be efficiently solved using standard QP solvers, such as *OSQP* [14].

Received 1 July 2025; revised 4 November 2025; accepted 8 December 2025. Date of publication 15 January 2026; date of current version 13 February 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62461160313, Grant 62273196, and Grant 62192752 and in part by the BNRist Project under Grant BNR2024TD03003. This article was recommended for publication by Associate Editor A. Del Prete and Editor P. M. Wensing upon evaluation of the reviewers' comments. *(Corresponding author: Yilin Mo.)*

The authors are with the Department of Automation and BNRist, Tsinghua University, Beijing 10084, China (e-mail: yangx21@mails.tsinghua.edu.cn; wangrun24@mails.tsinghua.edu.cn; luyw20@mails.tsinghua.edu.cn; ylmo@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TRO.2026.3653775

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

While this hierarchical approach ensures constraint satisfaction and optimizes performance, selecting an appropriate simplified dynamics model typically requires significant expertise and inevitably leads to deviation from the full model in long horizon prediction [4]. Furthermore, the finite planning horizon and model approximation may result in suboptimal performance and reduced robustness to perturbations, restricting the robot's full capabilities. The high computational cost of solving the MPC optimization problem online also limits the achievable control frequency, constraining the robot's agile and dynamic motion capabilities.

RL: In RL, the control policy, usually parameterized by a neural network, is learned directly through interactions with the environment [15], [16], [17]. Recent advances in GPU-accelerated simulators [18] have significantly boosted policy training efficiency by enabling massive parallel data collection [19]. In addition, sim-to-real techniques, such as domain randomization and adaptation [20] strengthen policy robustness by incorporating measurement noise, external disturbances, and model mismatches during training, leading to improved real-world generalization. In the training process, RL optimizes the policy parameters to maximize the expected cumulative reward, which generally includes task-specific rewards, penalties for constraint violations, and regularization terms. However, effective reward function design and hyperparameter selection often require extensive trial-and-error experimentation [21].

During deployment, the learned policy requires only a forward pass through the neural network, ensuring computational efficiency. However, constraints are incorporated only as soft penalties in the reward function rather than explicit constraints, offering no strict guarantees of constraint satisfaction. Studies have identified fundamental vulnerabilities in state-of-the-art neural controllers under adversarial attacks [22], highlighting the limitations of non-interpretable neural networks. Consequently, neither optimality nor safety can be guaranteed, restricting the applicability of RL in safety-critical and high-performance scenarios.

Our approach to combining OC with RL: In this work, we seek to integrate the complementary strengths of OC and RL for quadrupedal locomotion control during the online deployment phase. OC ensures *instantaneous constraint satisfaction* and *optimality* through explicit constraint satisfaction and objective minimization, while RL enables *robust* performance through extensive data-driven training with domain randomization. To leverage these advantages, we propose a hierarchical control framework where RL generates high-level reference trajectories and OC determines optimal motor torque commands through rigorous optimization, as demonstrated in Fig. 1(d).

A fundamental challenge of the proposed framework has been achieving scalable execution of OC during policy training, as RL desires simulating massive numbers of parallel instances to learn robust behaviors. To address this, we design a convex QP solver based on the Chambolle–Pock (CP) algorithm [23], which has proven to enable efficient parallel execution of constrained optimization on GPUs [24]. The proposed control framework is validated through extensive experiments, demonstrating its effectiveness in achieving safe, robust, and OC of legged robots.

A. Contributions

This work advances quadrupedal locomotion control through a hierarchical solution combining the merits of OC and RL, ensuring instantaneous constraint satisfaction, optimality, and robustness. Our key contributions are as follows.

1) We propose a hierarchical control framework that combines policy learning with optimization-based whole-body control, bridging the gap between the two approaches through a principled integration of their complementary strengths. The framework demonstrates improved tracking accuracy, energy efficiency, constraint satisfaction, and robustness compared to end-to-end RL and pure whole-body control methods.
2) We develop a convex quadratic programming (QP) solver based on first-order optimization that achieves both efficient parallel execution on GPUs and fast single-instance computation on CPUs. By decomposing each iteration into affine transformations and convex projections, our solver enables concurrent optimization across multiple environments while maintaining computation speed suitable for high-frequency control applications. The solver and its open-source implementation can be of independent interest to researchers from learning, control, and general robotics communities.
3) We enable efficient policy learning by integrating large-scale parallel optimization into the RL pipeline. This integration reduces parameter tuning complexity through explicit constraint handling, allows lower policy update frequencies through reliable tracking control, and facilitates transfer across different quadrupedal platforms through its hierarchical structure.
4) We validate the proposed framework through comprehensive simulations and hardware experiments on multiple quadrupedal platforms, with systematic evaluation of tracking performance, energy efficiency, and constraint satisfaction. The video is available at https://xu-yang16.github.io/rl_wbc_project/.

B. Outline

The rest of this article is organized as follows. Section II reviews related work, with emphasis on the computational bottlenecks limiting existing hybrid RL+OC methods. Section III introduces the CP algorithm based convex QP solver, explaining why its algorithmic structure enables efficient GPU parallelization and *how* it achieves both large-scale training and real-time deployment. Section IV then presents our hierarchical control framework that leverages this efficient solver to integrate the high-level RL policy with low-level whole-body controller. The solver's performance is benchmarked in Section V, demonstrating its advantages in both batched parallelization and single-instance real-time performance. Section VI validates the framework through comprehensive evaluations on both internal design choices and comparisons with existing methods. Section VII provides balanced perspective by analyzing limitations and tradeoffs. Finally, Section VIII concludes this article.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

II. Related Work

Our work advances the integration of RL with OC for legged locomotion through efficient optimization, safety guarantees, and scalable architecture. This section examines related work in these areas, beginning with existing approaches to combining RL and OC, followed by methods for ensuring safety in learning-based control, and concluding with GPU-accelerated optimization techniques that enable practical implementation of these methods.

A. Combining OC and RL in Legged Locomotion

The integration of OC with RL represents a key research direction in legged robot control. While early approaches focused on training RL policies to replicate trajectories from offline OC optimization [38], [39], recent work has explored architectures that combine OC and RL for online control. As illustrated in Fig. 1(c) and (d), existing methods can be categorized into two primary architectural patterns based on the hierarchical arrangement: OC $\rightarrow$ RL (OC provides reference trajectories for RL to track or refine) and RL $\rightarrow$ OC (RL provides high-level guidance to OC).

OC $\rightarrow$ RL Architecture: In this paradigm, OC provides reference trajectories or desired states that RL policies learn to track or compensate for. This approach typically involves a high-level planner (usually MPC) generating optimal trajectories using simplified dynamics models, while RL handles the complexities of tracking these trajectories with the full robot dynamics and addresses uncertainties arising from model mismatch and external disturbances. RL+VHIPM [25] exemplifies this approach, where a variable-height inverted pendulum model generates reference trajectories that are tracked by an RL policy. DTC [26] extends this concept by using RL to learn robust tracking controllers for NMPC-generated references, demonstrating superior performance on challenging terrains. RL policies can also learn residual control inputs [40] that are added to nominal control commands to achieve faster convergence and improved robustness. However, since RL directly optimizes the control input, sophisticated reward function design is typically required to balance desired task performance, energy efficiency, and safety.

RL $\rightarrow$ OC architecture: In this paradigm, RL operates at a higher level to provide distilled guidance to lower level modules. This approach can be further subdivided based on the type of information RL provides to the optimization layer.

1) *Reference generation:* Generalizable quadrupedal locomotion in diverse environments with a centroidal model (GLiDE) [29], continuous adaptive jumping with a learned centroidal policy (CAJun) [41], and corrective RL [30] employ RL to learn centroidal dynamics policies that generate desired center-of-mass trajectories, which are subsequently tracked by model-based controllers. In addition, RL can determine footstep placement [27], [42] and gait selection [43], with model-based controllers handling execution.
2) *Cost function learning:* Rather learning state references, this category involves learning cost function parameters for the optimization layer, including objective weights [32] and terminal costs [31] in MPC formulations.
3) *Dynamics model:* RL outputs dynamics parameters for more accurate modeling [34].
4) *Optimization initialization:* Considering the multiple local minima inherent in nonconvex problems, methods, such as continuous actor-critic with trajectory optimization (CACTO) [36] and its extension [37] learn RL policies to provide warm-start initializations for trajectory optimization solvers, accelerating convergence, and guiding toward superior solutions.

Nevertheless, the lower level OC module in this architecture can be either an MPC controller or a reactive WBC controller, both of which can provide formal safety guarantees that are challenging to achieve with pure RL approaches or OC$\rightarrow$ RL approaches.

Despite these advances, the primary bottleneck in implementing RL-OC hierarchies lies in the computational demands of parallel OC execution during policy training, as RL typically requires thousands of parallel environments for sample-efficient learning. As shown in Table I, previous work has circumvented this challenge through various compromises: employing simplified simulation environments [28], [29], severely limiting parallel training instances [25], [30], reducing OC update frequency [26], or approximating OC solutions [41]. These compromises often result in either reduced sample efficiency, compromised control performance, or both. Our approach directly addresses this fundamental limitation through a CP-based convex QP solver [23], enabling efficient parallel execution of rigorous OC during RL training without sacrificing control performance or safety guarantees. Furthermore, we provide a comprehensive analysis comparing different interface designs between RL and OC layers, including reference generation, cost function learning, and dynamics model learning approaches in Section VI-B. This contribution bridges the gap between the theoretical advantages of RL-OC integration and their practical implementation in large-scale parallel training scenarios.

B. Safe Legged Locomotion Control

Safety in the context of legged locomotion control refers to the ability to satisfy constraints in deployment, including joint torque limits, friction cone constraints, and contact force constraints, etc. The safety for training is not considered due to the fact that the training is generally done in simulation.

Despite its success in learning locomotion skills, RL faces fundamental challenges in safety-critical tasks due to the unconstrained nature of neural policies. Common RL approaches for legged locomotion transform hard constraints into soft penalty terms [19], [44], often requiring tedious hyperparameter tuning. To mitigate this limitation, constrained RL methods typically enforce feasible regions through either modifying the optimization criteria or restricting exploration spaces during the training process [45].

The first category incorporates constraints directly into the optimization criterion. Inspired by the Lagrangian method, primal-dual approaches [46] introduce Lagrangian multipliers and performs primal-dual updates at each policy iteration. Trust-region methods [47] limit policy updates to safe regions by constraining the step size. Penalized proximal policy optimization (P3O) [48],

TABLE I
COMPARISON WITH RELATED RL+OC METHODS

Method	RL Role	Full Physics	Parallel Envs	Training Time	OC:RL Frequency
VHIPM+RL [25]	Tracking Controller	×	1	46h	50:50
DTC [26]	Tracking Controller	✓	4096	336h	2.2:50
RLOC [27]	Tracking Controller	✓	12	32h	400:400
BG Policy [28]	Reference	×	12	1h	500:50
GLiDE [29]	Reference	×	1600	1-2h	100:100
Corrective RL [30]	Reference	✓	12	8.4h	400:400
DMPC [31]	Cost function	×	1	n/a	n/a
AC-MPC [32]	Cost Function	×	64	11.5h	50:50
AC4MPC [33]	Cost Function and Initialization	×	n/a	n/a	10:10
RL augmented MPC [34]	Dynamics Model	✓	1	n/a	33:33
MoPAC [35]	Dynamics Model and Initialization	✓	1	2h	n/a
CACTO [36], [37]	Initialization	✓	15	6.3h	100:100
Ours	Reference	✓	≥4096	0.5h	500:50

OC:RL Frequency indicates the frequency of OC module and RL policy updates. We use red to highlight the compromises made for the concurrent execution of OC and RL.

[49] and interior-point optimization methods [50], [51] enforce constraint satisfaction by introducing linear cost penalty terms or logarithmic barrier functions, respectively. The second category focuses on safe exploration strategies. One popular approach introduces another safety backup policy [52], [53], [54], enabling the robot to switch between nominal and recovery controllers when constraints might be violated. Another approach treats constraint violations as termination conditions [55], [56], [57], thereby restricting the robot to explore in safe regions.

Nevertheless, while constraints are incorporated during the training phase, there are no strict guarantees that the learned policy will respect constraints during deployment in real-world scenarios. Even during training, these methods generally provide only probabilistic constraint satisfaction guarantees. Moreover, they often struggle with the exploration process and exhibit unstable training behavior [58]. Our hierarchical control framework addresses these fundamental challenges by completely separating the problem: an upper level RL policy optimizes task performance in an unconstrained action space, while a lower level WBC enforces safety constraints through optimization at every control step. This architecture not only simplifies policy learning by eliminating the need for constraint handling in the RL formulation, but also maintains strict safety guarantees through the WBC layer during both training and deployment.

C. GPU-Accelerated Optimization

GPU-accelerated optimization has gained significant research attention with the rapid advancement of GPU parallelization capabilities. Existing approaches can be categorized into three main strategies: hardware-level acceleration, problem structure exploitation, and algorithm-level acceleration. *Hardware-level acceleration* leverages GPU's inherent parallel computing capabilities through vectorized and single instruction multiple data (SIMD) architectures. Representative works include *cuOSQP* [59], a GPU version of the popular QP solver *OSQP* [14], and CusADi [60], a code-generating framework for GPU parallelization of symbolic expressions. However, these methods typically require low-level CUDA programming to achieve optimal performance. *Problem structure exploitation* methods leverage the specific sparsity patterns and structures in the problem formulation, particularly in MPC applications. For example, Cholesky factorization of condensed systems can be accelerated using GPU [61]. GATO [62] enables real-time simultaneous solving of tens to hundreds of problems through codesigned parallelism tailored to the structure of trajectory optimization computations. Primal-dual iLQR methods [63] focus on equality-constrained problems and exploit temporal parallelization through parallel associative scans. These methods achieve impressive performance gains but are typically limited to specific problem classes with known structural properties. *Algorithm-level acceleration* designs optimization algorithms specifically for massive parallelization, fundamentally reformulating the solution process to achieve the highest level of generality, free from dependencies on specific hardware architectures or problem structures. A notable example is rectified linear unit (ReLU)-QP [64], which reformulates the alternating direction method of multipliers algorithm into multiple layers of ReLU functions for efficiently solving large-scale optimization problems. Beyond single large problems, research has also explored developing batched solvers for multiple small optimization problems. The *qpth* solver [65], for instance, utilizes a primal-dual interior point method and achieves substantial acceleration through batch matrix factorizations and prefactorization of invariant Karush-Kuhn-Tucker (KKT) conditions. Detailed comparisons can be found in Table II and Section V.

In this article, we propose a convex QP solver capable of handling convex set constraints with analytical projections, representing a novel contribution in the algorithm-level acceleration category. The solver is specifically designed for two purposes: massive parallel execution of batched instances on GPUs, making it particularly suitable for concurrent RL training, and efficient deployment on CPUs for real-time control. Our solver achieves these capabilities by leveraging the CP method, which decomposes each optimization iteration into only two operations—affine transformations and convex projections. As demonstrated in our benchmarks, this approach delivers high computational efficiency in both training and deployment.

TABLE II
COMPARISON WITH RELATED QP SOLVERS

Method	GPU Solver	Batched Solver	Constraints	Algorithm	Order	Warm Start
OSQP	×	×	Linear	Augmented Lagrangian	1	✓
ReLU-QP [64]	✓	×	Linear	Augmented Lagrangian	1	✓
cuOSQP [59]	✓	×	Linear	Augmented Lagrangian	1	✓
CuClarabel [67]	✓	×	Conic	Interior Point	2	×
jaxopt-cvxqp [68]	✓	×	Linear	Interior Point	2	✓
qpax [69]	✓	✓	Linear	Interior Point	2	×
qpth [65]	✓	✓	Linear	Interior Point	2	×
jaxopt-osqp [68]	✓	✓	Linear	Augmented Lagrangian	1	✓
jaxproxqp [70]	✓	✓	Linear	Augmented Lagrangian	1	✓
Ours	✓	✓	Convex	CP	1	✓

III. CONVEX QP SOLVER BASED ON CP METHOD

In this section, we present a solver based on the CP algorithm to solve optimization problems with quadratic objectives and convex set constraints, commonly referred to as convex QP or general QP problems in the literature [14]. These problems can be formulated as

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}x^\top Qx + p^\top x \tag{1a}$$

$$\text{s.t.} \quad Hx + b \in \mathcal{C} \tag{1b}$$

where $x \in \mathbb{R}^n$ is the decision variable, and the quadratic objective function is determined by a matrix $Q \in \mathbb{S}^n_{++}$ and the vector $p \in \mathbb{R}^n$. The matrix $H \in \mathbb{R}^{m\times n}$, the vector $b \in \mathbb{R}^m$, and the convex set $\mathcal{C} \subseteq \mathbb{R}^m$ together define the constraints. Here, n, m are the dimensions of the decision variable and the constraint, respectively, and $\mathbb{S}^n_{++}$ denotes the set of $n \times n$ positive definite matrices. Specifically, when $\mathcal{C}$ is a polyhedron, the problem reduces to a standard QP problem.

We make the following assumptions throughout this article.

1) $Q \in \mathbb{S}^n_{++}$ is a strictly positive definite matrix.
2) The constraint set $\mathcal{C}$ is a nonempty, closed, and convex set, whose projection operator $\text{Proj}_{\mathcal{C}}(x) \triangleq \underset{y\in\mathcal{C}}{\text{argmin}}\|x-y\|_2^2$ can be evaluated analytically.

We first rewrite problem (1) as an equivalent form by introducing a new variable $z \in \mathbb{R}^m$

$$\begin{aligned}\min_{x\in\mathbb{R}^n, z\in\mathbb{R}^m} \quad & \frac{1}{2}x^\top Qx + p^\top x \\ \text{s.t.} \quad & Hx + b = z \\ & z \in \mathcal{C}.\end{aligned} \tag{2}$$

This reformulation separates the quadratic objective and linear equality constraints from the convex set constraint, enabling us to leverage efficient projection operations in the solution algorithm. Using the indicator function of the set $\mathcal{C}$, defined as

$$\mathbb{I}_{\mathcal{C}}(z) \triangleq \begin{cases} 0, & \text{if } z \in \mathcal{C} \\ +\infty, & \text{otherwise} \end{cases} \tag{3}$$

we can turn problem (2) into the following form:

$$\min_{z\in\mathbb{R}^m} f(z) + g(z) \tag{4}$$

where the functions $f(z)$ and $g(z)$ are defined as

$$f(z) \triangleq \mathbb{I}_{\mathcal{C}}(z) \tag{5a}$$

$$g(z) \triangleq \{\min_x \frac{1}{2}x^\top Qx + p^\top x \mid Hx + b = z\}. \tag{5b}$$

The function $f(z)$ encodes the constraint set $\mathcal{C}$, while $g(z)$ can be evaluated by solving a convex quadratic program with only linear equality constraints. This decomposition separates the convex set constraint from the quadratic objective, enabling us to leverage the structure of each term in our solution approach.

Using the Fenchel duality theory [66, Ch. 9], we can formulate the dual problem of (4) as

$$\max_{\lambda\in\mathbb{R}^m} -f^*(-\lambda) - g^*(\lambda) \tag{6}$$

where f^* and g^* are the Fenchel conjugates of f and g, respectively, and $\lambda \in \mathbb{R}^m$ is the dual variable. Furthermore, the Lagrangian of the primal-dual problem pair can be written as

$$L(\lambda, z) = f(z) + \lambda^\top z - g^*(\lambda). \tag{7}$$

Theorem 1: The optimal solution $(\lambda^\star, z^\star)$ to problem (7) can be obtained by iteratively performing the following two operations as long as $\alpha > 0, \beta > 0$, and $\alpha\beta < 1$.

1) Perform the primal-dual update

$$\begin{bmatrix}\lambda^{k+1} \\ z^{k+1}\end{bmatrix} \leftarrow A\begin{bmatrix}\lambda^k \\ z^k\end{bmatrix} + B \tag{8}$$

where A and B are defined as

$$A \triangleq \begin{bmatrix} F & \beta F \\ \alpha(I-2F) & I - 2\alpha\beta F\end{bmatrix}, B \triangleq \begin{bmatrix}\beta\mu \\ -2\alpha\beta\mu\end{bmatrix}$$

$$F \triangleq (I + \beta HQ^{-1}H^\top)^{-1}, \mu \triangleq F(HQ^{-1}p - b).$$

2) Project z^{k+1} onto the convex set $\mathcal{C}$

$$z^{k+1} \leftarrow \text{Proj}_{\mathcal{C}}\left(z^{k+1}\right). \tag{9}$$

With the above update rules, we have: $\lim_{k\to\infty} z^k = z^\star$ and $\lim_{k\to\infty}\lambda^k = \lambda^\star$.

Proof: The proof is provided in Appendix A.

If needed, we can finally recover the optimal solution $x^\star$ to the original problem (1) by solving the following KKT equation:

$$Qx^\star + H^\top\lambda^\star + p = 0. \tag{10}$$

The CP algorithm can be understood as a "predict-correct" method. At each iteration, we first make a prediction step using

an affine transformation that incorporates gradient information from both primal and dual variables, then correct any constraint violations through projection onto the feasible set. This intuition translates directly to our two-step update rule in Theorem 1: (8) performs the prediction via matrix multiplication, while (9) applies the correction through projection. This two-step process naturally separates the optimization dynamics from constraint handling. This approach is particularly valuable for robotics because it provides a systematic way to handle hard constraints through efficient projection operations, and upon convergence, guarantees satisfaction of instantaneous safety-critical constraints, such as joint limits and friction cones.

Interestingly, while our solver is rigorously grounded in optimization theory, it exhibits a striking structural resemblance to a multilayer perceptron (MLP): each iteration consists of a linear transformation (8) followed by a nonlinear activation function [the projection operation in (9)], mirroring the architecture of an MLP layer. As long as the projection can be evaluated inexpensively on GPUs, the solver is able to run in massive parallel, which is particularly suitable for quadrupedal locomotion control training tasks, as shown in Section IV-A.

As an example, we consider the following convex QP problem:

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}x^\top \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix} x + \begin{bmatrix} -0.5 \\ -0.4 \end{bmatrix}^\top x$$
$$\text{s.t.} \quad x \in \{x \mid x_1^2 + x_2^2 \le 1\}. \tag{11}$$

Fig. 2 shows the convergence trajectory toward $x^\star = [0.8; 0.3]$ and $\lambda^\star = [0; 0]$ for different parameters α and β (we keep $\alpha\beta$ fixed to 0.81). Geometrically, the affine transformation moves the solution based on gradient information (shown as solid lines), while the projection operation enforces feasibility in the transformed space by pulling any constraint-violating point back into the feasible region (shown as dashed lines), with convergence guaranteeing satisfaction of the original constraints. The hyperparameters α and β control the update step size of the primal variables x and the dual variables λ, respectively. Large value of α tends to cause frequent violation of the constraint and the solution x oscillates around the boundary of the constraint, while small value of α leads to slow convergence. The same conclusion can be drawn for β from the convergence trajectory of λ. Proper selection of these parameters can balance between the primal and dual updates, significantly improving the convergence speed.

Discussion on ill-posed and infeasible problems: For ill-conditioned problems with poor convergence, we can integrate the Ruiz equilibration heuristic used in OSQP to improve the condition number. Specifically, we apply diagonal scaling matrices to transform the problem before applying our CP iterations. For infeasibility detection, we monitor the difference between consecutive iterates [71]. Infeasible primal problems are detected when $\|\lambda^{k+1} - \lambda^k\|$ fails to converge to zero, while infeasible dual problems are indicated by nonconvergent $\|z^{k+1} - z^k\|$. In practice, we implement early termination when these residuals exceed predefined thresholds after a maximum number of iterations.

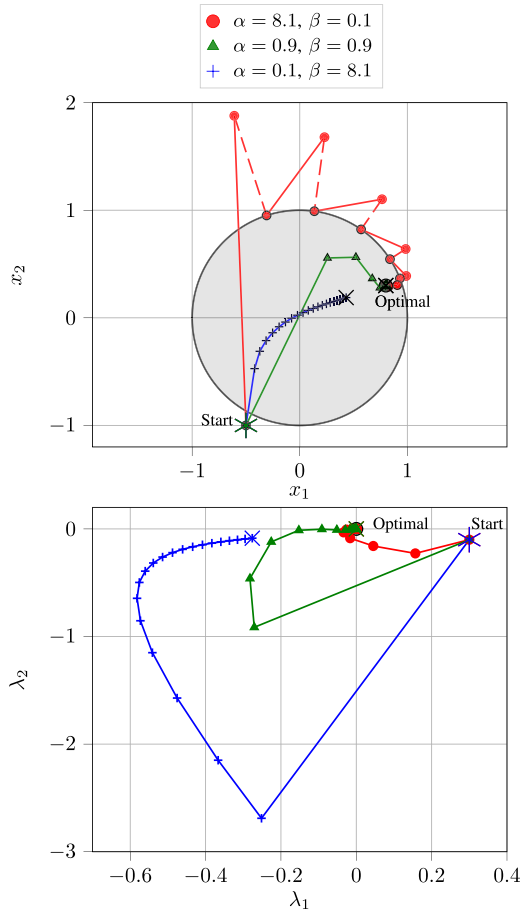

Fig. 2. Convergence trajectory of primal variable x and dual variable λ for different α and β with 20 iterations. The feasible region is shaded in gray. Affine transformation is marked with solid lines, while projection is marked with dashed lines.

The comparison in Table II highlights several key advantages of our proposed solver over existing approaches. Most existing GPU-based QP solvers, including ReLU-QP and cuOSQP, lack batched processing capabilities, limiting their efficiency when solving multiple QP problems simultaneously. While these solvers excel at large-scale individual problems, they cannot leverage the parallelism benefits that batched operations provide. Among the solvers that do support batching, our approach offers distinct computational advantages. Interior-point methods like qpth and qpax, despite their theoretical second-order convergence, require expensive matrix factorizations at each iteration, particularly challenging in batched scenarios where multiple factorizations must be performed simultaneously. In contrast, our first-order CP algorithm avoids these costly operations while maintaining competitive convergence properties. The constraint handling capabilities further differentiate our solver. While most existing methods are restricted to linear constraints, our approach extends to general convex constraints through analytical projections. This flexibility enables broader applicability, particularly in robotics and control applications where complex feasible regions are common. The combination of convex constraint support with efficient batched processing represents a significant advancement over current linear-constraint-only batched solvers. Our proposed solver also provides the additional feature

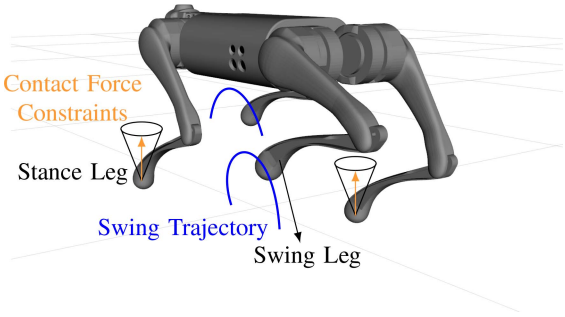

Fig. 3. Illustration of stance and swing legs during quadrupedal locomotion.

of warm-starting with a guessed primal and dual solution, which proves particularly valuable in locomotion control scenarios, where temporal consistency allows previous optimal solutions to serve as high-quality initial guesses for subsequent time steps, as empirically demonstrated in Section V.

IV. Hierarchical Control and Learning Architecture

This section presents a hierarchical control architecture that integrates RL with whole-body control. We formulate the whole-body control (WBC) problem for quadrupedal locomotion as a convex quadratic program and develop a hierarchical framework combining high-level RL policies with low-level WBC optimization. The proposed CP solver enables efficient parallel policy training through its capacity for massively parallel optimization.

A. WBC Problem Formulation

The instantaneous whole-body control of quadruped robots can be formulated as a trajectory optimization problem. The objective is to find optimal joint torques that drives the robot to follow a reference trajectory while respecting the constraints.

Depending on their contact states during locomotion, legs of quadrupedal robots can be divided into two categories: stance legs and swing legs, which is illustrated in Fig. 3. The stance legs maintain contact with the ground and support the robot's body, while the swing legs are in the air and move to the next contact point. As highlighted in prior research [5], the key to stable legged locomotion is the control of ground reaction forces (GRFs) for stance legs. In contrast, the control of the swing legs generally involves merely following a predetermined trajectory [72].

To minimize the tracking error between the actual and desired base accelerations, reduce power consumption, avoid sliding of stance legs, and comply to the joint torque limits, we formulate the optimization problem as

$$\min_{\tau} \quad \|\ddot{q}_{\text{b, actual}} - \ddot{q}_{\text{b, ref}}\|_R^2 + \|\tau\|_S^2 + \|\dot{q}^\top \tau\|_T^2 \tag{12a}$$

$$\text{s.t.} \quad D(q, \dot{q}, \ddot{q}, f) = 0 \qquad \text{(dynamics constraint)} \tag{12b}$$

$$f_{i,\text{world}} \in \mathcal{K}_{\text{fric}} \qquad \text{for stance legs} \tag{12c}$$

$$f_{i,z,\text{world}} \in [0, z_{\max}] \qquad \text{for stance legs} \tag{12d}$$

$$f_i = 0 \qquad \text{for swing legs} \tag{12e}$$

$$\tau \in [-\tau_{\max}, \tau_{\max}] \qquad \text{for all motors} \tag{12f}$$

where the objective function consists of the velocity tracking error, the heat power, and the mechanical power, $\tau \in \mathbb{R}^{12}$ represent joint torques, and $\ddot{q}_{\text{b, actual}} \in \mathbb{R}^6$, $\ddot{q}_{\text{b, ref}} \in \mathbb{R}^6$ denote the actual and desired base accelerations, respectively. The weighting matrices are defined as $R \in \mathbb{R}^{6\times6}$, $S \in \mathbb{R}^{12\times12}$, and $T = tI^{1\times1}$, where t is a scalar. The generalized position, velocity, and acceleration are represented by q, $\dot{q}$, and $\ddot{q}$, respectively. The GRFs of all legs in the base frame are represented as $f \triangleq [f_1; f_2; f_3; f_4] \in \mathbb{R}^{12}$. The GRF of the ith leg in the world frame is denoted as $f_{i,\text{world}} = R^{\text{base}}_{\text{world}} f_i$, which can be obtained with the rotation matrix $R^{\text{base}}_{\text{world}}$. The vertical contact force for the ith leg is denoted with $f_{i,z,\text{world}} \in \mathbb{R}$. The friction cone, $\mathcal{K}_{\text{fric}}$, is defined as

$$\mathcal{K}_{\text{fric}} = \{f_i \in \mathbb{R}^3 \mid \|f_{i,xy}\|_2 \le \mu f_{i,z}\}$$

where μ is the friction coefficient and $f_{i,xy}$ represents the horizontal component of the contact force. Note that the friction cone $\mathcal{K}_{\text{fric}}$ can be approximated [73] by a friction pyramid, replacing second-order cone constraints with linear constraints

$$\mathcal{P}_{\text{fric}} = \{f_i \in \mathbb{R}^3 \mid f_{i,x}, f_{i,y} \in [-\mu f_{i,z}, \mu f_{i,z}]\}. \tag{13}$$

Considering the light-weighted legs of quadruped robots, the whole-body dynamics of the robot is often approximated with simplified dynamics, accelerating the construction and computation of the QP problem. One common choice is the SRBD model

$$\ddot{q}_b = M_{\text{inv}}(q) f + g_b \tag{14}$$

where the matrix $M_{\text{inv}} \in \mathbb{R}^{6\times12}$ is determined by the base mass, inertia, and foot positions, and the vector $g_b \in \mathbb{R}^6$ is constant. The detailed derivation can be found in Appendix B.

Since the joint torques and GRFs are determined by the dynamics equation $\tau = -J(q)^\top f$ with the Jacobian matrix $J(q)$, we can rewrite the optimization problem by regarding the GRFs f as the decision variables.

By solving $\ddot{q}_{\text{b, actual}}$ from the dynamics equation (14) and substituting it into the optimization problem, we obtain the following convex QP problem:

$$\min_{f} \quad \frac{1}{2} f^\top Q f + p^\top f \tag{15a}$$

$$\text{s.t.} \quad f_{i,\text{world}} = R^{\text{base}}_{\text{world}} f_i \in \mathcal{K}_{\text{fric}} \qquad \text{for all legs} \tag{15b}$$

$$f_{i,z,\text{world}} = [0\ 0\ 1] R^{\text{base}}_{\text{world}} f_i \in [0, z_{\max,f}] \qquad \text{for all legs} \tag{15c}$$

$$\tau = -J(q)^\top f \in [-\tau_{\max}, \tau_{\max}] \qquad \text{for all motors} \tag{15d}$$

where the filtered maximum force

$$z_{\max,f} \triangleq \begin{cases} z_{\max}, & \text{for stance legs} \\ 0, & \text{for swing legs} \end{cases} \tag{16}$$

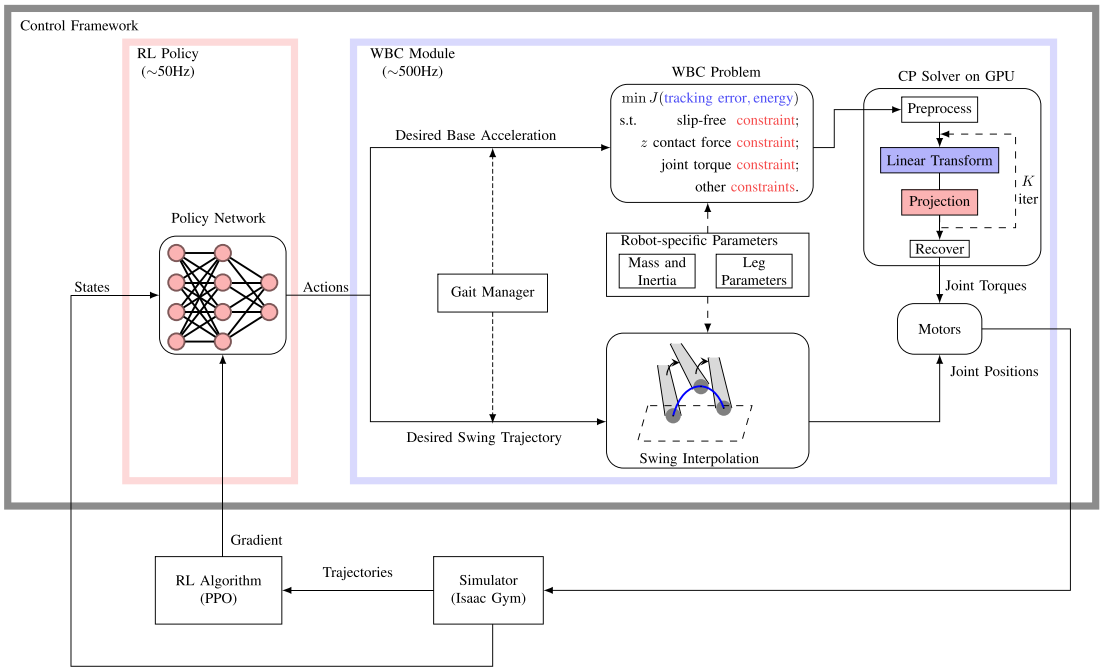

Fig. 4. Proposed training framework with end-to-end GPU execution. All components run on GPU, enabling massive parallelization across thousands of environments. The RL policy runs at a relatively low frequency, generating swing trajectories for swing legs and desired base acceleration, which is safely and efficiently tracked by a high-frequency WBC controller. The WBC problem is solved by our proposed GPU-accelerated convex QP solver, allowing the entire control pipeline to execute in parallel during policy training. The gait manager module determines the gait phases and the contact states of the legs. Robot-specific parameters are directly fed into the WBC problem and the swing trajectory generator without the need of training.

is used to unify the constraints for stance and swing legs, and $Q = M_{\text{inv}}^{\top} R M_{\text{inv}} + J(S + t\dot{q}q^{\top})J^{\top}$ and $p = M_{\text{inv}}^{\top} R^{\top}(g_b - \ddot{q}_{b,\text{ref}})$ are the cost function parameters.

In problem (15), only three types of constraints are involved: *second-order conic (SOC)* constraints, *interval* constraints, and *positive orthant* constraints, which can be handled in a unified manner using our proposed convex QP solver. We present the standard forms of these constraints and provide the analytical expression for the projection operator each type of constraint.

1) *SOC constraint:* $f_i \in \mathcal{K}_{\text{soc}}$, where $\mathcal{K}_{\text{soc}} = \{(f_{i,xy}, f_{i,z}) \in \mathbb{R}^{2+1} \mid \|f_{i,xy}\|_2 \le f_{i,z}\}$. With $v \triangleq f_{i,xy}$ and $s \triangleq f_{i,z}$, the projection can be expressed as

$$\Pi_{\mathcal{K}_{\text{soc}}}(f_i) = \begin{cases} 0, & s \le 0 \\ f_i, & \|v\|_2 \in [0, |s|] \\ \left(\frac{1}{2} + \frac{s}{2\|v\|_2}\right) \begin{bmatrix} v \\ \|v\|_2 \end{bmatrix}, & \|v\|_2 > |s|. \end{cases} \tag{17}$$

2) *Interval constraint:* $f_i \in [z_{\min}, z_{\max}]$. The projection into the interval $[z_{\min}, z_{\max}]$ can be expressed as

$$\Pi_{[z_{\min}, z_{\max}]}(f_i) = \max(z_{\min}, \min(z_{\max}, f_i)). \tag{18}$$

3) *Positive orthant constraint:* $f_i \in \mathbb{R}_+$. The projection onto the nonnegative orthant is

$$\Pi_{\mathbb{R}_+}(f_i) = \max(0, f_i). \tag{19}$$

With the pyramid approximation for friction cone, a single *SOC* constraint is reduced to two *interval* constraints, which can also be handled by the solver. The WBC formulation can also be extended to include additional convex constraints without affecting the solver's effectiveness, as long as the projection operation is analytically available.

B. Training Framework

The proposed controller consists of a neural RL policy module operating at 50 Hz coupled with a high-frequency (500 Hz) whole-body controller. The WBC module serves as an integral component of the control architecture during both training and deployment—it executes optimized joint torques based on the base acceleration commands generated by the neural policy, as illustrated in Fig. 4. This integration enables the policy to learn commands that naturally account for the WBC's optimization constraints and dynamics. The timescale separation between policy and control frequencies (50 Hz versus 500 Hz) allows efficient learning while maintaining precise low-level control through high-frequency optimization.

In RL, the control problem is typically modeled as a Markov decision process, which is defined by the state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $R(s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the state transition probability $p(s_{t+1}|s_t, a_t)$. The goal is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$, mapping from states to actions, to maximize the expected sum of discounted rewards over time described by the return $G_t = \sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k})$, where $\gamma \in (0, 1)$ is the

discount factor that balances potential future rewards and immediate rewards. The optimal policy π^* maximizes the expected return from each state

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s_{t+1}\sim p(s_t,\pi(s_t))}\left[\sum_{k=0}^{\infty}\gamma^k R(s_{t+k}, a_{t+k})\right]. \tag{20}$$

In our proposed training framework, the neural policy is executed in a simulation environment, where the state s_t is fed into the policy network, generating the action a_t. Depending on the gait phases, stance legs and swing legs are controlled differently. The actions $a_t \in \mathbb{R}^{12}$ include the desired base acceleration $\ddot{q}_{\text{b, ref}} \in \mathbb{R}^6$ and the foot position residues $\Delta x_{\text{foot}} \in \mathbb{R}^6$ for swing legs. We first construct the optimization problem to minimize tracking errors and energy consumption while respecting all necessary constraints. We use the proposed convex QP solver for both massive parallel running in training and fast execution in single robot deployment. The WBC module synchronizes all motors, determining joint torques to economically track the base acceleration. For swing legs, the residue Δx_{foot} is used to adjust the swing height and the landing xy positions, determining the swing trajectory. The joint torque and positions obtained above are fed to the motors in the simulators, which then updates the robot states and generates the rewards for the neural policy to learn. The policy is trained using the proximal policy optimization (PPO) algorithm [74] to maximize the expected return. Specific definitions of the states and reward functions are discussed below.

The states $s_t \in \mathbb{R}^{55\times N}$ of the robot include historical observations over a time window N. At each time step, the instantaneous observation can be conceptually divided into three components: *desired commands* $\in \mathbb{R}^3$, *gait phases* $\in \mathbb{R}^8$, and *proprioceptive observations* $\in \mathbb{R}^{44}$. For the velocity tracking task, the desired commands consist of the forward velocity $v_{x,\text{ref}}$, lateral velocity $v_{y,\text{ref}}$, and yaw rate $\omega_{z,\text{ref}}$. The gait observations $(\sin(\phi), \cos(\phi))$ are derived from the current gait phase ϕ. The proprioceptive observations are defined as the robot's roll $\psi \in \mathbb{R}$, pitch $\theta \in \mathbb{R}$, base velocity $v_{\text{b}} \in \mathbb{R}^3$, base angular velocity $\omega_{\text{b}} \in \mathbb{R}^3$, foot positions $p_{\text{foot}} \in \mathbb{R}^{12}$, foot velocities $v_{\text{foot}} \in \mathbb{R}^{12}$, and last action $a_{t-1} \in \mathbb{R}^{12}$.

In our framework, RL policies are only responsible for the base pose stabilization and velocity tracking task, agnostic of leg parameters. Therefore, the reward function only involve three terms.

1) *Base pose stabilization:* $r_{\text{sta}} = -0.1(h_{\text{actual}} - h_{\text{ref}})^2 - 0.06(\psi^2 + \theta^2) - 0.001(\dot{\psi}^2 + \dot{\theta}^2)$. The height h_{actual} and h_{ref} are the actual and desired height of the robot's base, respectively.
2) *Velocity tracking:* $r_{\text{vel}} = 0.002\exp\{-4\|v_{xy,\text{actual}} - v_{xy,\text{ref}}\|_2^2\} + 0.001\exp\{-4(\omega_{z,\text{actual}} - \omega_{z,\text{ref}})^2\} - 0.02v_{z,\text{actual}}^2$.
3) *Action smoothing:* $r_{\text{smooth}} = -10^{-5}\|a_t - a_{t-1}\|_2^2$.

This reward setting, free of penalties or terms related to specific physical parameters, benefits from the hierarchical architecture, where low-level details are handled by the WBC module, significantly simplifying the reward tuning and policy learning process.

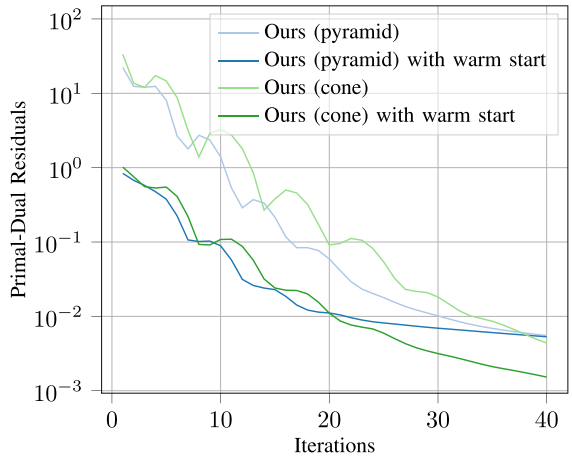

Fig. 5. Convergence of the proposed solver versus iterations.

V. Solver Benchmark

In this section, we benchmark the proposed convex QP solver on the aforementioned WBC problem for quadrupedal locomotion tasks. The evaluation focuses on three key metrics: *convergence performance*, *time consumption*, and *parallel capability*, highlighting the solver's efficiency in both batched execution and real-time deployment. Beyond empirical performance, we analyze the *algorithmic characteristics* that enable these capabilities—providing insights for future algorithm designers seeking to develop GPU-friendly optimization methods. We have implemented the proposed solver in Python with NumPy and PyTorch, and the code is publicly available.[1] Note that the solver and its implementation can be of independent interest to researchers from learning, control, and general robotics communities.

The benchmark is conducted on a computer with a single NVIDIA GeForce RTX 4090 GPU and 3.7 GHz AMD Ryzen 9 5900X CPUs. The primary focus for deployment is on the solver's speed and convergence performance, while for batched execution, we emphasize time consumption and scalability. We compare our proposed convex QP solver against the widely recognized *OSQP* solver [14], which is extensively utilized in real-time control tasks. In addition, we include comparisons with existing batch QP solvers, including *qpax* [69], *jaxproxqp* [70], *jaxopt-osqp* [68], and *qpth* [65].

In deployment involving a single robot, we benchmark our proposed solver, which incorporates pyramid and conic constraints, against the *OSQP* solver. Fig. 5 illustrates the primal-dual residuals versus iterations, calculated as $\|Hx + b - z\|_2 + \|Qx + p - H^\top\lambda\|_2$. The residuals decrease to 1% of their initial value within 20 iterations, adequately stabilizing the robot in simulation through the WBC controller. For comparative analysis, Fig. 6 evaluates the relative error from the optimal solution of *OSQP*, defined as $\|x_{\text{ours}} - x_{\text{osqp}}\|_2/\|x_{\text{osqp}}\|_2$. Our solver achieves convergence to the optimal solution of *OSQP*

[1][Online]. Available: https://github.com/xu-yang16/cp_solver

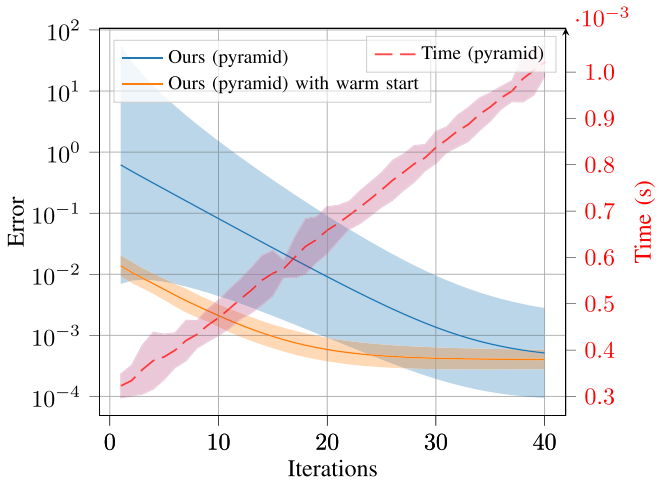

Fig. 6. Relative error from the optimal solution of *OSQP* and time consumption versus iterations (4000 environments). The shaded region represents the standard deviation.

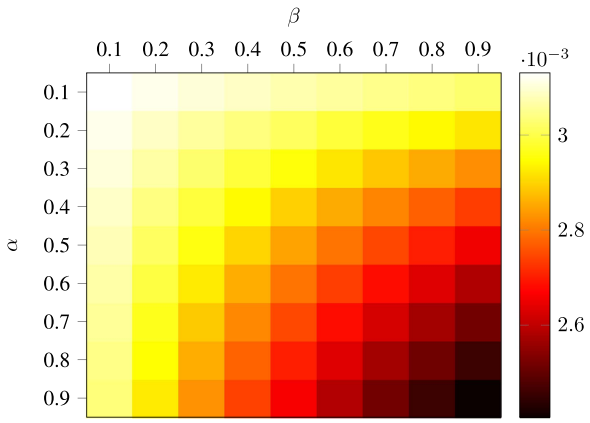

Fig. 7. Relative error (compared with *OSQP*) heatmap of different α and β values with warm start (20 iterations).

TABLE III
REAL-TIME SPEED OF SOLVERS FOR DEPLOYMENT

Solver	Frequency (Hz)
qpth	106.07±15.95
OSQP	1369.08±151.63
Ours (pyramid)	**1390.51±102.42**
Ours (cone)	1183.20±61.45

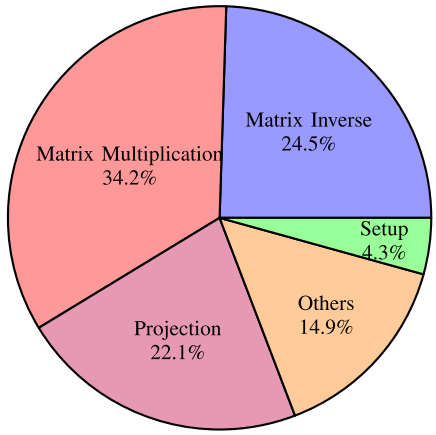

Fig. 8. Computational time breakdown of the CP solver with NumPy.

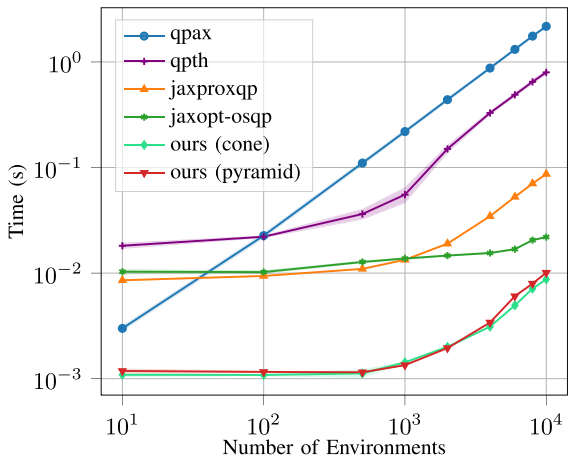

Fig. 9. Time consumption of the proposed solver versus number of environments. The shaded region represents the standard deviation.

within 20 iterations, with an error margin of less than 0.1%. In addition, employing a warm start significantly enhances the convergence speed, benefitting from the WBC controller's high operating frequency of approximately 500 Hz. We also present the relationship between iteration count and computational runtime in Fig. 6, revealing an essentially linear correlation, where 40 iterations require approximately 1ms of execution time. To evaluate the algorithm's hyperparameter sensitivity, we perform experiments with a WBC controller for quadruped locomotion using various paris of α and β parameters. The corresponding relative errors are depicted in Fig. 7, demonstrating that while the product $\alpha\beta$ marginally affects convergence rate in this task, the overall impact remains negligible.

The real-time performance of the solver is detailed in Table III. We compare the NumPy implementation of our proposed solver with *OSQP* and *qpth*, which are implemented in C and PyTorch, respectively. The results indicate that our solver slightly outperforms *OSQP* in handling the WBC problem with pyramid constraints and significantly surpasses *qpth*. As shown in Fig. 8, our solver's computational time is dominated by dense matrix operations (80.8% in matrix multiplication, inverse, and projection), which are highly optimized in NumPy's underlying BLAS libraries. In contrast, *OSQP* spends over 90% of its time in preprocessing steps, including converting matrices to sparse formats to accelerate iterative linear solves. For our small-scale dense problems ($n = 12$ and $m = 32$), this preprocessing overhead outweighs the benefits of sparse computation during iteration. There is potential for further enhancing the solver's performance through optimization in C or CUDA.

For batched execution, we employ a PyTorch-based implementation of our proposed solver. Fig. 9 demonstrates the solver's capability to perform efficiently in massively parallel environments, resulting in significant computational efficiency

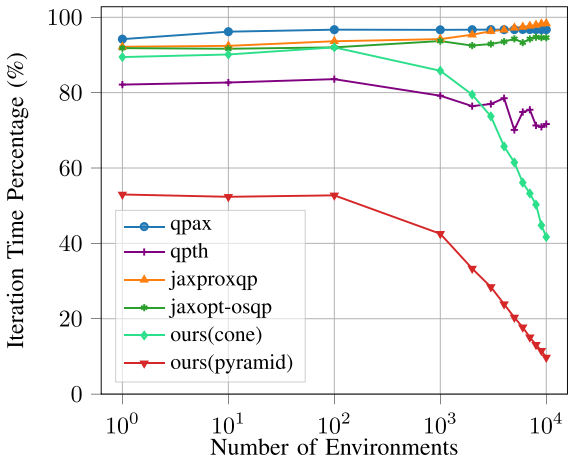

Fig. 10. Iteration time percentage of different solvers, i.e., the proportion of total computation time spent in the core iterative solve loop versus overhead operations (memory allocation, matrix precomputation, etc.). Lower percentages at larger batch sizes indicate better parallelization efficiency.

gains. Despite the just-in-time compilation and vectorized mapping capabilities of the JAX framework, our solver achieves approximately 5.16× speedup compared to *jaxopt-osqp* and demonstrates a remarkable 109.6× acceleration over the second-order interior point method *qpth* when evaluated across 4000 parallel environments.

Fig. 10 provides compelling evidence for why the CP algorithm is inherently well-suited for GPU acceleration. This figure shows the percentage of total computation time spent in the core iterative solve loop as the batch size increases from 1 to 10 000 environments. Our proposed solver (both cone and pyramid variants) exhibits dramatically decreasing iteration time percentages as batch size increases—from approximately 50% at single-instance to below 10% at 10 000 environments (for the pyramid version). In stark contrast, competing solvers like qpax, qpth, and jaxproxqp maintain iteration percentages above 70%–95% even at large batch sizes. This difference reveals a fundamental architectural advantage: our solver's computational bottleneck shifts from the iterative solve itself to fixed overhead [memory allocation, one-time F matrix computation in (8), etc.] as parallelism increases, indicating near-perfect scaling of the core algorithm with batch size. The key distinction is that interior point methods (*qpax*, *qpth*) require repeated matrix factorizations within each iteration, while augmented Lagrangian methods (*jaxproxqp*, *jaxopt-osqp*) involve nested iterative linear system solves at every outer iteration—both introducing variable-complexity operations that resist GPU parallelization. In contrast, our first-order solver eliminates all inner loops by precomputing the matrix F once, transforming each iteration into fixed-cost operations (matrix-vector multiplications and element-wise projections) that map directly to highly optimized GPU BLAS kernels. These operations exhibit excellent cache locality and enable regular, predictable memory access patterns that maximize GPU memory bandwidth utilization.

Table IV presents the profiling results of the proposed training framework. The policy training takes approximately 30 min of wall-clock time, with the WBC controller running at 500 Hz and the RL policy at 50 Hz. During the training, the time consumption for the WBC updates is comparable to that of the physics simulations, indicating that solving the QP online is no longer a bottleneck in the training process.

TABLE IV
PROFILING FOR EACH POLICY ITERATION (ABOUT 3 S OF WALL-CLOCK TIME)

Terms	WBC Update	Forward Sim	Policy Update	Others
Percent	39.7%	37.1%	4.9%	18.4%

VI. EXPERIMENTS

In this section, we conduct comprehensive evaluations to validate the effectiveness of our proposed hierarchical control framework. We design our experiments in two phases: first, we perform ablation studies within the proposed framework to justify our design choices; second, we compare against state-of-the-art model-free RL and hybrid RL+OC methods to demonstrate the advantages of our approach. The rest of this section is organized as follows: We first introduce the experimental setup and then present the ablation studies within the RL+WBC framework. Finally, we compare our method with existing model-free and hybrid RL+optimization methods.

A. Experimental Setup

Policy training is conducted in the Isaac Gym [18] simulator on the same computer used for solver benchmarking. For sim-to-real deployment, both policy inference and WBC computations are executed on an Intel Core i7-13620H processor equipped with 32 GB of RAM. We perform sim-to-sim validation using the MuJoCo [75] simulator before deploying policies on a Unitree Go1 [76] quadruped robot to evaluate real-world performance. The training code is available online[2] and the experiment video is available at the project website.[3]

B. Ablation Study on RL Output

To validate our design choice of learning reference trajectories rather than other control components, we conduct comprehensive ablation studies comparing different settings within the RL+WBC framework along with a classical convex MPC+WBC method.

1) *MPC+WBC:* An SRBD-based convex MPC controller [5] is used to generate reference trajectories for the WBC controller.
2) *Ours (reference):* RL policy outputs desired base acceleration $\ddot{q}_{b,\text{ref}} \in \mathbb{R}^6$ and swing foot position residuals $\Delta x_{\text{foot}} \in \mathbb{R}^3$.
3) *Ours (cost):* RL policy learns adaptive cost function weights for the matrix R [in (12)] in the WBC optimization and swing foot position residuals Δx_{foot}. We consider R as a diagonal matrix and only learn the diagonal 6 elements.

[2][Online]. Available: https://github.com/xu-yang16/rl_wbc
[3][Online]. Available: https://xu-yang16.github.io/rl_wbc_project/

4) *Ours (dynamics):* RL policy learns residual corrections to the centroidal dynamics model parameters, including the robot mass m and a diagonal inertia matrix $I_b \in \mathbb{R}^{3\times3}$ and swing foot position residuals Δx_{foot}.

For fair comparison, all variants use identical network architectures, reward functions, training procedures, and the same CP-based WBC solver. Policies are trained for 600 epochs ($\sim$ 30 min) on flat ground velocity tracking. Both MPC and RL policies operate at 50 Hz with a fixed 2 Hz gait frequency. Safety constraints include maximum contact force of 100 N and joint torque of 12 N·m. Note that relaxing these constraints would expand the feasible region, allowing faster tracking responses and higher accuracy.

To precisely control experimental conditions, eliminate confounding factors, and accurately measure physical quantities that are not readily accessible in the real world, we conduct velocity tracking experiments in the MuJoCo simulation environment, with results presented in Fig. 11. In contrast to the flat terrain used during training, the experiment is performed on rough terrain, which simultaneously validates the policy's robustness to unknown external disturbances. As indicated by the desired forward velocity trajectory marked with a black dashed line, the robot is initially commanded to remain stationary for the first 2 s, followed by tracking a reference forward velocity of 0.8 m/s for the remainder of the experiment. In addition, from the fourth second onwards, a 3 kg mass (about 25% of the robot mass) is attached to the robot's base to simulate a model mismatch scenario. We set the maximum vertical contact force to 100 N, which is slightly above the two-feet standing contact force of 65 N, and the maximum joint torque to 12 N·m as safety constraints, highlighted by red dashed lines in the figure.

First, all methods effectively respect the predefined constraints. The contact force remains well below the 100 N limit across all RL+WBC variants. Joint torque constraints are consistently satisfied, with all methods operating within the $\pm$ 12 N·m bounds, demonstrating the effectiveness of the QP formulation in enforcing safety limits.

For velocity tracking, under nominal conditions ($t < 4$ s), all three RL+WBC variants—learning reference, cost, and dynamics—demonstrate comparable and accurate velocity tracking performance, closely following the desired trajectory. In contrast, the classical MPC+WBC approach shows slightly inferior tracking accuracy. This degradation stems from the inherent limitations of MPC on rough terrain, where early or late contact reduce the accuracy of future state predictions based on the dynamics model, leading to suboptimal reference generation. When the payload perturbation is introduced at $t = 4$ s, these methods exhibit distinct adaptive behaviors.

1) *Ours (reference)* maintains similar tracking performance by learning to generate significantly modified desired base accelerations to compensate for the changed dynamics.
2) *Ours (cost)* achieves velocity tracking by adaptively adjusting the velocity tracking cost weights in the WBC optimization, though the adaptation response is slightly slower compared to reference learning.
3) *Ours (dynamics)* shows the largest performance degradation, likely because the learned dynamics corrections.

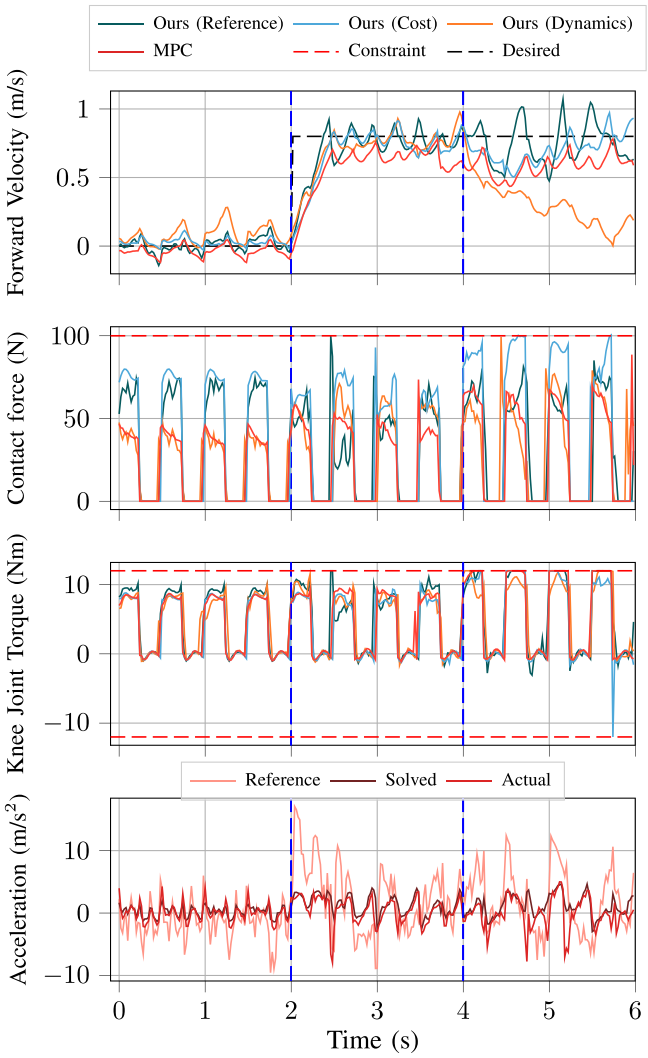

Fig. 11. Commanded velocity tracking experiment on rough terrain in MuJoCo. (a) Actual and desired forward velocity. (b) Vertical force of the front right (FR) foot. (c) Torque of the FR knee joint. (d) For *Ours (Reference)* policy, we show the reference acceleration (i.e., policy output), solver acceleration given by the WBC module and the actual acceleration.

4) *MPC+WBC* exhibits substantial steady-state tracking error after the perturbation. This degradation occurs because the model mismatch further compromises the accuracy of future state predictions, leading to increasingly suboptimal trajectory planning.

As shown in Fig. 11, we plot the reference acceleration (i.e., the output of the RL policy), solved acceleration (the best-effort tracking result within feasible regions), and actual acceleration under disturbances and model mismatch circumstances. The WBC layer incorporates the robot's intrinsic dynamics, which enables excellent alignment between the solved and actual accelerations, demonstrating the effectiveness of our physics-informed optimization approach.

The mismatch between reference and solved accelerations reveals two important aspects of our framework. First, it illustrates how safety constraints actively limit the feasible region, necessarily compromising raw tracking performance to

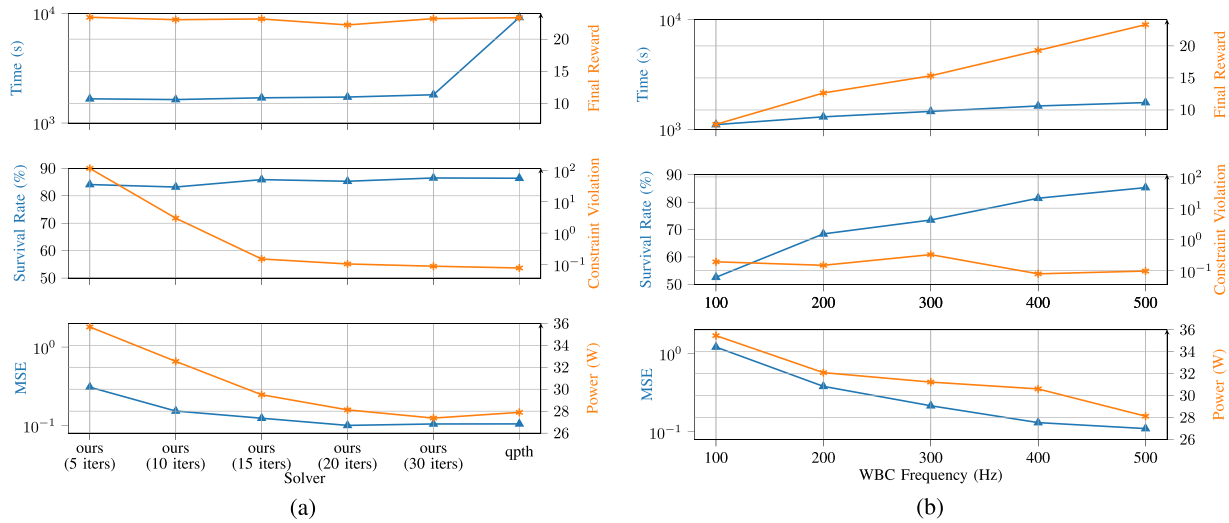

Fig. 12. Ablation studies on WBC configurations. For each configuration, results are averaged over 5 independently trained policies with different random seeds. (a) Ablation study on solver iterations. (b) Ablation study on WBC frequency.

ensure constraint satisfaction—a fundamental tradeoff between performance and safety. Second, and more significantly, it demonstrates the remarkable adaptability of our learned policy. Through extensive training with domain randomization, the RL policy has developed the capability to provide robust guidance even when its commands cannot be executed exactly. Despite the constraint-induced mismatch at the acceleration level, the policy's strategic planning enables the robot to still achieve accurate velocity tracking at the task level, showcasing the hierarchical framework's ability to maintain high-level objectives while respecting low-level physical limitations.

C. Ablation Study on WBC Configuration

We further conduct ablation studies on the solver configuration and the low-level WBC frequency to validate their impact on the overall learning and control performance. We consider the following variants:

1) *Solver accuracy:* We compare different numbers of solver iterations (5, 10, 15, 20, 30) against the *qpth* baseline to investigate the tradeoff between computational efficiency and solution accuracy.
2) *WBC frequency:* We evaluate WBC update frequencies of {100 Hz, 200 Hz, 300 Hz, 400 Hz, 500 Hz} to examine the impact of low-level control on overall task performance.

Each configuration is evaluated across 5 random seeds with identical training hyperparameters. The policy is evaluated in 1000 virtual environments for 15 s in the Isaac Gym simulator, with all environments following the same reference velocity trajectory and suffers from the same disturbances. The metrics compared include survival rate, constraint violation magnitude, velocity tracking mean-squared error (MSE), and mean mechanical power consumption, where the mechanical power is calculated as $p = \sum_{i=1}^{12} |\tau_i \dot{q}_i|$. To assess training efficiency, we compare the final reward achieved after 600 epochs of training for each configuration along with corresponding time consumption.

As shown in Fig. 12(a), increasing solver accuracy (i.e., more iterations) leads to significant improvements in constraint satisfaction, task performance, and energy efficiency. Notably, with only 20 iterations, our solver achieves performance comparable to *qpth*, while being over 5 times faster during training. Interestingly, the final RL reward and robustness (i.e., survival rate) do not improve significantly with solver accuracy, indicating that an RL policy can partially compensate for inaccurate low-level control through learned adaptive behavior.

Similarly, Fig. 12(b) illustrates that higher WBC frequencies yield nearly monotonic improvements across all performance metrics. Notably, increasing the WBC frequency from 100 to 500 Hz results in a substantial 66.0% reduction in velocity tracking MSE and a 20.7% decrease in mechanical power consumption, highlighting the critical role of high-frequency low-level control in enhancing both task performance and energy efficiency. Moreover, the survival rate increases from 52.59% to 85.30%, demonstrating that higher control frequencies significantly improve the robot's ability to maintain stability and satisfy constraints under disturbances.

D. Comparison With Existing Methods

Building upon the ablation studies, we now compare our best-performing approach (i.e., RL outputs reference tracked by the WBC module) against existing methods, including model-free, optimization-based method, and established hybrid RL+optimization methods. We set the solver iteration to 20 and WBC frequency to 500 Hz for our method based on the ablation results in Section VI-C.

1) *End-to-end RL policy* [44]: Direct mapping from states to desired joint positions, which is specifically trained on low-friction environments (the friction coefficient is set to 0.2–0.5).

2) *Constrained RL policy:* A first-order constrained RL algorithm, N-3PO [48], [49], is used to train a policy that penalizes the constraint violations as linear penalties in the reward function, which is recommended through a comparative study on legged locomotion [48]. The constraints include the following:
   1) vertical contact forces no larger than $100N$;
   2) joint torque no larger than 12 N·m;
   3) no foot slip.
3) *RL+MPC* [32]: RL policy provides cost function weights (with dimension 12 balancing pose stabilization and velocity tracking tasks) for MPC solver (50 Hz). We employ the classical convex MPC [5] with a prediction horizon 0.5 s (one gait cycle) and timestep 0.025 s. The constraints above are incorporated in the QP formulation of MPC, which is solved with *OSQP*. The reward setting is the same with ours.
4) *Ours:* Our proposed method with RL generating reference trajectories for WBC tracking (500 Hz). The same constraints are incorporated in the QP formulation solved with the proposed convex QP solver.

All methods employ policy networks comprising three hidden layers with sizes of 512, 256, and 128, respectively, and the policy frequency is set to 50 Hz. The gait frequency is manually fixed to 2 Hz. We conduct parallel simulation using 4096 robots simultaneously on rough terrain for data collection and policy update. However, due to the computational limitations of *OSQP* in large-scale parallelization, the RL+MPC baseline is trained with only 32 environments and the training takes roughly 8 h.

We evaluate the proposed training framework on a flat ground commanded velocity tracking task for quadrupedal robots, highlighting the following merits of our method.

1) *Training:* Elimination of extra rewards and, thus, reduced parameter tuning; accelerated convergence in policy training and enhanced sample efficiency.
2) *Task performance and constraint satisfaction:* Optimized velocity tracking accuracy, energy efficiency, and guaranteed enforcement of constraints.
3) *Robustness:* Resilience against external disturbances and model mismatch.
4) *Generalization:* Transferability to various quadruped robots with the same RL policy.

RL policy training: The reward setting is detailed in Table V, where $\times$ represents the necessity of the reward term. The rewards are divided into stabilization, task, smoothing, and penalty categories, addressing the objectives of stability, task performance, control input refinement, and the avoidance of undesired or unsafe behaviors, respectively. In contrast to the end-to-end and constrained RL methods, which requires numerous terms to meticulously regulate the movement of swing legs and motors, our approach concentrates solely on the base state and GRFs, eliminating the need for additional penalty or auxiliary rewards, which facilitates policy transferability to different quadrupedal robots, as demonstrated in the subsequent generalization analysis. This merit of our method stems from the hierarchical structure of the control framework and the centralized control of all actuators handled by the WBC module.

TABLE V
COMPARISON OF REWARD TERMS

Category	Rewards	Approaches		
		Ours	E2E RL	N-3PO
Stabilization	Body Orientation	×	×	×
	z Velocity	×	×	×
	Roll Pitch Velocity	×	×	×
Task	Velocity Tracking	×	×	×
	Height Tracking	×	×	×
Smoothing	Action Smoothing	×	×	×
Penalty	Stance Leg Velocity		×	
	Swing Leg Force		×	×
	Footswing xy		×	
	Footswing Height		×	
	Foot Slip		×	×
	Collision		×	×
	Joint Torques		×	×
	Joint Velocities		×	×
	Joint Accelerations		×	
	Foot Symmetry			×
	Stand Still			×
	Contact Force			×

For constrained RL methods (i.e., N-3PO), the cost term coefficient is also needed to be tuned, therefore it is included as a reward term in this comparison.

TABLE VI
COMPARISON OF THE PROPOSED METHOD WITH BASELINES ACROSS DIFFERENT METRICS

Policy	MSE	Power (W)	Survival Rate (%)
Ours	**0.11±0.01**	**28.12±1.03**	85.3±2.1
RL+MPC	0.17±0.05	34.75±3.47	80.2±4.5
MPC+WBC	0.23±0.02	49.15±1.62	62.3±4.0
End-to-end RL	0.42±0.01	82.96±1.17	**100.0±0.0**
Constrained RL	0.47±0.05	56.37±2.03	75.3±2.4

Best is highlighted in **bold** and second best is underlined.

Following the same experimental setup with Section VI-C, we further compare our hierarchical framework against baselines in terms of velocity tracking accuracy, energy efficiency, and robustness. As shown in Table VI, our approach achieves the lowest tracking MSE and power consumption, demonstrating that the WBC layer significantly enhances both tracking precision and energy efficiency compared to purely learning-based methods.

Task performance and constraint satisfaction:

Fig. 13 presents the vertical contact force validation experiments across three scenarios. The results demonstrate that our RL+WBC framework successfully maintains force constraints below the specified threshold across the following three scenarios:

1) traversing fragile foam blocks;
2) navigating highly compliant deformable terrain;
3) operating under real-time decibel monitoring.

The decibel meter provides quantitative verification of constraint satisfaction, confirming that the WBC module effectively enforces safety limits during dynamic locomotion. The decibel meter provides quantitative verification of vertical contact forces, confirming that the WBC module effectively enforces safety limits during dynamic locomotion. On flat ground, our

(a) (b) (c)

Fig. 13. Real-world experiments on vertical contact force constraints. With our control framework, the robot is commanded to traverse fragile, deformable terrain. We set the maximum vertical force to 80 N in the constraint formulation. Real-time experimental videos are available online. (a) Robot traverses fragile terrain. (b) Robot navigates through deformable terrain. (c) To quantify the contact force, a decibel meter is used to evaluate the vertical contact force.

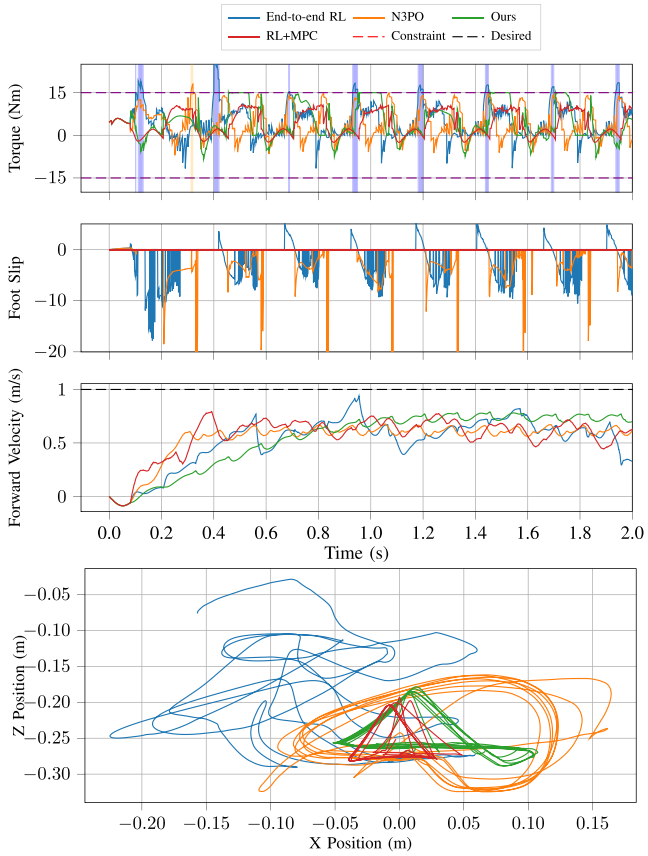

Fig. 14. Commanded velocity tracking experiment on slippery surfaces demonstrating constraint satisfaction and task performance. (a) FR knee joint torques with constraint violation highlighted. (b) Foot slip. (c) Velocity tracking performance. (d) Foot trajectory in the robot frame.

method achieves a maximum decibel level of 68.1 dB, while RL+MPC records 70.3 dB, demonstrating similar performance. In contrast, end-to-end RL and constrained RL exhibit maximum decibel levels of 88.1 dB and 76.7, respectively, indicating significantly higher contact forces.

As shown in Fig. 14, we conduct a commanded velocity tracking experiment on slippery surfaces. The joint torque, foot slip (defined as the foot velocity multiplied by the foot contact force, which is estimated by joint torques and the foot Jacobian matrix along with the pneumatic foot pads of the robot), and velocity tracking curves are shown in the figure. The results demonstrate that both our method and RL+MPC successfully satisfy the joint torque limits (15 N·m) and no-slip constraints. In addition, both approaches achieve stable velocity tracking performance. Due to the extremely low surface friction coefficient (μ <0.15), precise velocity tracking becomes challenging as the GRFs are severely limited. Under such conditions, our method demonstrates the smallest steady-state tracking error compared to the baseline approaches. It is notable that both methods exhibit an extended acceleration phase at the beginning of the experiment. This behavior stems from the fact that our WBC layer consistently operates within the feasible constraint region, ensuring safety at the cost of reduced tracking aggressiveness. The tradeoff manifests as a more conservative acceleration profile where the low-level WBC prioritizes constraint satisfaction over rapid reference tracking, resulting in slower but safer convergence to the desired velocity. This demonstrates the inherent safety-performance tradeoff in our hierarchical framework, where constraint enforcement takes precedence over tracking speed when operating near the limits of the robot's physical capabilities.

Robustness: To unleash the full capacity of our framework, we perform robustness experiments with all hard constraints disabled. By removing these constraints, we create an unconstrained QP problem where the reference trajectory has full control authority over the actuators. Such experimental flexibility is enabled by the transparent and modular nature of our WBC component, allowing real-time modification of constraint formulations and demonstrating that our framework's robustness extends beyond mere constraint satisfaction. The robustness of our framework is attributed to the robustness of the learned policy by virtue of domain randomization during training.

We conducted 10 trials for each policy on challenging terrains as shown in Fig. 15, including slopes, obstacles, and uneven ground. The experimental results demonstrate that our method successfully completed 7 out of 10 trials, while end-to-end RL achieved 10 out of 10 successful completions, constrained RL completed 9 out of 10 trials, and RL+MPC failed to complete any of the trials (0 out of 10). These results highlight the tradeoffs between constraint satisfaction and task performance: while our method prioritizes strict constraint enforcement, which may limit performance in highly demanding scenarios, it provides guaranteed instantaneous constraint satisfaction compared to methods that treat constraints as soft penalties.

Fig. 15. Our RL+WBC framework enables the robot traversing diverse challenging terrains including slopes, obstacles, and uneven ground.

TABLE VII
COMPARISON OF OUR POLICY TRAINED WITH GO1 (A SINGLE POLICY FOR ALL ROBOTS), RL+MPC TRAINED WITH GO1 (A SINGLE POLICY FOR ALL ROBOTS), MPC+WBC, AND END-TO-END RL METHODS

Robot	Policy	MSE	Power (W)	Survival Rate (%)
A1	Ours	**0.12±0.01**	**40.18±0.60**	88.6±0.8
	RL+MPC	0.23±0.05	42.10±0.61	81.2±2.5
	MPC+WBC	0.30±0.03	50.22±1.09	73.4±0.4
	E2E RL	0.42±0.01	90.67±0.07	**100.0±0.0**
Go2	Ours	**0.15±0.01**	**53.79±0.75**	90.8±2.0
	RL+MPC	0.20±0.02	42.10±0.61	81.2±2.5
	MPC+WBC	0.25±0.02	55.20±1.40	76.3±2.1
	E2E RL	0.41±0.01	73.00±0.07	**99.2±0.7**
Mini Cheetah	Ours	**0.20±0.01**	**40.71±0.83**	92.3±1.9
	RL+MPC	0.23±0.05	42.10±0.61	83.2±2.5
	MPC+WBC	0.27±0.02	47.93±3.68	78.7±3.8
	E2E RL	0.38±0.01	61.00±0.10	**98.9±0.9**

Best is highlighted in **bold**, and second best is underlined.

Transferability: We evaluate the transferability of the proposed control framework across various quadruped robots by conducting the same experiments outlined in Table VI with the Unitree A1, Go2, and MIT mini cheetah robots in the Isaac Gym simulator. The results, presented in Table VII, compare the performance of the same RL policy trained on the Go1 robot against other methods (end-to-end RL policy is retrained from scratch). The proposed control framework consistently outperformed the alternatives in tracking accuracy and power consumption across all robots, while also maintaining a high survival rate.

This success in transferability can be attributed to the hierarchical structure of our framework, which conceals the intricacies of each robot's dynamics and kinematics behind the WBC controller and provides a uniform high-level interface for the policy network. Consequently, the RL policy in our framework operates in a robot-agnostic action space consisting of desired base accelerations and foot position residuals, instead of learning to control joint torques or positions which requires retraining for each specific robot model. Similarly, the RL+MPC method also allows transferability to other similar robots since the MPC controller handles robot-specific dynamics.

Failure cases: We also show failure cases that primarily occur in two scenarios. First, overly restrictive constraints can limit the robot's feasible action space, preventing necessary recovery maneuvers when aggressive responses are required for maintaining stability. This performance degradation stems from our WBC formulation operating within the feasible region defined by safety constraints, which intentionally limits aggressive actions that violate torque limits, contact force or friction constraints. However, this conservative behavior represents a principled design choice rather than a limitation. The constraints ensure that the robot never exceeds safe operating parameters, as demonstrated in our real-world experiments where model-free RL policy caused visible terrain damage while our method successfully preserved both robot and environment integrity. The tradeoff between maximum agility and guaranteed constraint satisfaction makes our approach particularly valuable for applications requiring reliable, long-term operation under uncertain conditions.

Second, the predefined gait pattern restricts the robot's full degrees of freedom by rigidly defining stance and swing leg assignments, with many failure cases resulting from swing legs colliding with unexpected obstacles during their predetermined trajectories. Model-free RL methods typically achieve balance recovery through rapid switching between stance and swing legs, dynamically adapting the contact schedule based on environmental demands. In contrast, our current dependence on a fixed gait scheduler, while simplifying the controller design and ensuring predictable behavior, also reduces the robot's adaptive potential in highly dynamic or cluttered environments. We also include a discussion on existing limitations and possible future work in Section VII.

Analysis: The fundamental limitation of MPC-based methods lies in their reliance on accurate hybrid dynamics models: predefined gait patterns determine contact schedules spanning the prediction horizon, while dynamics models assume precise system parameters. This rigid structure renders them vulnerable to model mismatches and environmental uncertainties.

Conversely, the robustness of model-free RL approaches stems from their implicit ability to autonomously shift priorities—dynamically adjusting the relative importance of competing goals (task performance versus stabilization) based on learned experience, effectively implementing a context-dependent cost function. However, this adaptive balancing capability of RL inherently treats constraints as soft limitations that can be violated when necessary, such as sacrificing constraint satisfaction for improved task performance. This flexibility becomes critically dangerous in safety-critical applications where constraint violations are not tolerated.

In contrast, our proposed RL+WBC framework strictly confines all behaviors within the feasible region, ensuring absolute satisfaction of hard constraints. Admittedly, when constraints are particularly restrictive, this approach involves certain tradeoffs in task performance and robustness. Nevertheless, in scenarios where constraint violations are discouraged rather than strictly forbidden, we can enhance overall performance through constraint relaxation or even selective constraint abandonment since the constraints in the WBC layer can be dynamically adjusted.

VII. DISCUSSION

Our hierarchical RL+WBC framework involves several design tradeoffs that stem from separating high-level policy learning from low-level constraint enforcement. While these characteristics provide advantages in certain scenarios

(guaranteed constraint satisfaction, energy efficiency, cross-platform transferability), they also impose limitations in others.

1) *Predefined gait patterns:* Our current framework relies on predetermined gait sequences, limiting the robot's ability to discover novel locomotion strategies or adapt gait patterns dynamically to extreme terrains. This restriction simplifies controller design and ensures predictable behavior but reduces adaptive potential in highly dynamic environments. However, since the WBC formulation is solely based on the robot's current state and does not inherently depend on future foot contact states, we believe that the gait scheduler can be replaced with a neural policy, enabling learned gait adaptation while maintaining the safety guarantees of the optimization-based controller.
2) *Model dependency:* The WBC module relies on reasonably accurate dynamics models (e.g., SRBD models) and kinematic parameters, whereas end-to-end RL can potentially compensate for model uncertainties and unmodeled dynamics through learning. Furthermore, constraint parameters, although can be dynamically adjusted, need to be predetermined. Unrealistic constraint setting could cause overly conservative behavior or system failures.
3) *Performance tradeoffs from hard constraints:* While hard constraints ensure safety, they also restrict the robot's full actuation capacity. Our hierarchical structure guides the lower level WBC controller within predefined feasible regions. End-to-end RL, having full authority over all motors, may achieve superior performance in scenarios where constraint violations are temporarily acceptable.
4) *Scope of constraint guarantees:* Like other reactive control methods, our WBC cannot guarantee predictive constraints that require future state information. For instance, collision avoidance with obstacles requires predicting future trajectories and planning over time horizons, which our reactive controller does not provide.

Addressing these limitations while maintaining the advantages of our hierarchical approach motivates several promising directions for future work.

1) *Extension to higher degree-of-freedom systems:* Considering the low level WBC in our control framework can guarantee strict constraint satisfaction, task planning can be naturally incorporated through weighted or hierarchical formulations to achieve multiple task objectives simultaneously. Therefore, we anticipate extending our learning framework to control systems with higher degrees of freedom, such as quadrupedal robots equipped with manipulators or humanoid robots with dual arms. In contrast, end-to-end RL approaches require complex hyperparameter tuning to balance multiple tasks, and the exploration efficiency problem becomes increasingly pronounced as the action dimensionality increases. Our hierarchical framework could potentially address these challenges by enabling systematic task prioritization through optimization-based planning while maintaining efficient high-level policy learning.
2) *Adaptive gait generation:* Currently, the robot gaits remain predefined in our framework. A promising direction is to incorporate gait selection and adaptation into the RL policy, enabling autonomous gait transitions and adaptive gait adjustments in response to varying terrain conditions and task requirements. This extension would combine the adaptability of learned gait patterns with the precise and safe whole-body dynamic control provided by the lower level WBC, potentially achieving both versatility and energy efficiency in locomotion control.

VIII. Conclusion

In this article, we develop a hierarchical control framework for quadrupedal locomotion featuring a high-level RL policy and a low-level high-frequency whole-body controller, integrating the robustness of RL policies with the energy efficiency and instantaneous constraint satisfaction of OC. To allow the RL policy fully exploit the capability of robots, we design a convex QP solver based on the CP method, capable of both scaling to massive parallelization on GPU and fast real-time deployment and thus enabling concurrent training of RL policies and large-scale parallel running of whole-body control. Owing to the hierarchical structure, the RL policy can focus on high-level tasks, i.e., base acceleration reference and footstep planning, while the high-frequency whole-body controller ensures the robot's stability, instantaneous constraint satisfaction (joint torque limits, friction cones, contact forces etc.), and energy efficiency during locomotion. Extensive experiments in both simulation and real-world validate the merits on policy training, safety, and energy efficiency for deployment, robustness in unseen scenarios, and generalization in quadrupedal locomotion control.

Appendix

A. Proof of Theorem 1

Derivation of the update rule: For the Lagrangian function (7), the saddle differential is given by

$$\partial L(z,\lambda) = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix} \begin{bmatrix} z \\ \lambda \end{bmatrix} + \begin{bmatrix} \partial f(z) \\ \partial g^*(\lambda) \end{bmatrix} \tag{21}$$

whose fixed-point iteration [66] is

$$\begin{bmatrix} z^{k+1} \\ \lambda^{k+1} \end{bmatrix} = (I + \partial L)^{-1} \begin{bmatrix} z^k \\ \lambda^k \end{bmatrix}. \tag{22}$$

With the change of variables

$$\begin{bmatrix} y \\ w \end{bmatrix} = M \begin{bmatrix} z \\ \lambda \end{bmatrix} \tag{23}$$

and M is chosen as a positive definite matrix

$$M \triangleq \begin{bmatrix} \frac{1}{\alpha} I & I \\ I & \frac{1}{\beta} I \end{bmatrix}$$

we have

$$\begin{bmatrix} z^{k+1} \\ \lambda^{k+1} \end{bmatrix} = (M + \partial L)^{-1} M \begin{bmatrix} z^k \\ \lambda^k \end{bmatrix} = \left(\begin{bmatrix} \frac{1}{\alpha} I & 2I \\ 0 & \frac{1}{\beta} I \end{bmatrix} + \begin{bmatrix} \partial f \\ \partial g^* \end{bmatrix} \right)^{-1} \begin{bmatrix} \frac{1}{\alpha} z^k + \lambda^k \\ z^k + \frac{1}{\beta} \lambda^k \end{bmatrix}. \quad (24)$$

That is

$$\begin{bmatrix} \frac{1}{\alpha} I & 2I \\ 0 & \frac{1}{\beta} I \end{bmatrix} \begin{bmatrix} z^{k+1} \\ \lambda^{k+1} \end{bmatrix} + \begin{bmatrix} \partial f(z^{k+1}) \\ \partial g^*(\lambda^{k+1}) \end{bmatrix} \ni \begin{bmatrix} \frac{1}{\alpha} z^k + \lambda^k \\ z^k + \frac{1}{\beta} \lambda^k \end{bmatrix}. \quad (25)$$

By the definition of the proximal operator

$$\operatorname{Prox}_{\alpha f}(y) = \underset{z \in \mathbb{R}^m}{\operatorname{argmin}} \left\{ \alpha f(z) + \frac{1}{2} \|z - y\|^2 \right\} \quad (26)$$

we have

$$\begin{aligned} \lambda^{k+1} &= \operatorname{Prox}_{\beta g^*} \left(\lambda^k + \beta z^k \right) \\ z^{k+1} &= \operatorname{Prox}_{\alpha f} \left(z^k - \alpha \left(2\lambda^{k+1} - \lambda^k \right) \right). \end{aligned} \quad (27)$$

To compute $\operatorname{Prox}_{\beta g^*}$, we start from Moreau's identity [66]

$$\phi = \operatorname{Prox}_{\beta g^*}(\phi) + \beta \operatorname{Prox}_{g/\beta}(\phi/\beta) \quad \forall \phi \in \mathbb{R}^m, \beta > 0. \quad (28)$$

Therefore, we instead compute $\operatorname{Prox}_{g/\beta}(\phi)$

$$\begin{aligned} \operatorname{Prox}_{g/\beta}(\phi) &= \underset{z}{\operatorname{argmin}} \begin{Bmatrix} \min_x \frac{1}{2} x^\top P x + q^\top x + \frac{\beta}{2} \|z - \phi\|_2^2 \\ \text{s.t. } Hx + b = z \end{Bmatrix} \\ &= \underset{z}{\operatorname{argmin}} \begin{Bmatrix} \frac{1}{2} x^\top P x + q^\top x + \frac{\beta}{2} \|z - \phi\|_2^2 \\ + \nu^\top (Hx + b - z) \end{Bmatrix}. \end{aligned} \quad (29)$$

The KKT conditions of the equality-constrained QP problem are

$$\begin{bmatrix} P & 0 & H^\top \\ 0 & \beta I & -I \\ H & -I & 0 \end{bmatrix} \begin{bmatrix} x^\star \\ z^\star \\ \nu^\star \end{bmatrix} = \begin{bmatrix} -q \\ \beta\phi \\ -b \end{bmatrix}. \quad (30)$$

Since P is positive definite and $\beta > 0$, the left-hand side of the KKT system is invertible. By solving this linear system, we can obtain the explicit solution of $\operatorname{Prox}_{g/\beta}(\phi)$

$$\begin{aligned} \operatorname{Prox}_{g/\beta}(\phi) &= z^\star \\ &= \phi - \left(\beta H P^{-1} H^\top + I \right)^{-1} \left(\phi + H P^{-1} q - b \right) \\ &= \beta H \left(\beta H^\top H + P \right)^{-1} H^\top \phi - \mu \\ &= (I - F)\phi - \mu \end{aligned} \quad (31)$$

where $F \triangleq (\beta H P^{-1} H^\top + I)^{-1} = I - \beta H (P + \beta H^\top H)^{-1} H^\top$ and $\mu \triangleq F(H P^{-1} q - b)$.

By substituting $\operatorname{Prox}_{g/\beta}(\phi/\beta)$ into Moreau's identity (28), we can obtain the explicit solution of $\operatorname{Prox}_{\beta g^*}(\lambda)$

$$\operatorname{Prox}_{\beta g^*}(\lambda) = F\lambda + \beta\mu. \quad (32)$$

With the (27), we have

$$\lambda^{k+1} = F\lambda^k + \beta F z^k + \beta\mu. \quad (33)$$

Note that the proximal operator of the indicator function is the projection operation. Therefore, we have

$$z^{k+1} = \operatorname{Proj}_{\mathcal{C}} \left(z^k - \alpha \left(2\lambda^{k+1} - \lambda^k \right) \right). \quad (34)$$

In summary, the primal-dual update rule is given by cyclic executions of the matrix multiplication (8) and the projection operation (9).

Convergence analysis: Since all we need is to ensure the matrix M is positive definite, we have $\alpha > 0, \beta > 0$ and $\alpha\beta < 1$. The fixed point iteration of the primal-dual update rule (24) is guaranteed to converge to the optimal solution $(\lambda^\star, z^\star)$ of problem (7).

B. Derivation of the Centroidal Dynamics model (14)

With the assumption of massless legs, the robot is modeled as a rigid body with mass $m \in \mathbb{R}$ and inertia $I_\mathrm{b} \in \mathbb{R}^3$ (in the base frame). In the base frame, the centroidal dynamics of the robot without external disturbances is given by

$$\begin{bmatrix} m I_{3\times3} & 0 \\ 0 & I_\mathrm{b} \end{bmatrix} \ddot{q}_\mathrm{b} = \begin{bmatrix} \sum_{i=1}^4 f_i + R_\mathrm{base}^\mathrm{world} \begin{bmatrix} 0 & 0 & -mg \end{bmatrix}^\top \\ \sum_{i=1}^4 r_i \times f_i \end{bmatrix} \quad (35)$$

where $\ddot{q}_\mathrm{b}$ is the linear and angular acceleration of the robot, f_i is the contact force at the ith foot, r_i is the vector of the ith foot relative to the centroid, and g is the acceleration due to gravity. The centroidal dynamics can be rewritten as

$$\begin{aligned} \ddot{q}_\mathrm{b} &= \begin{bmatrix} 1/m I_{3\times3} & 0 \\ 0 & I_\mathrm{b} \end{bmatrix} \begin{bmatrix} \sum_{i=1}^4 f_i + R_\mathrm{base}^\mathrm{world} \begin{bmatrix} 0 & 0 & -mg \end{bmatrix}^\top \\ \sum_{i=1}^4 r_i \times f_i \end{bmatrix} \\ &= M_\mathrm{inv}(q) f + g_b \end{aligned} \quad (36)$$

where the matrix M_inv and the vector $g_b \in \mathbb{R}^6$ are defined as

$$\begin{aligned} M_\mathrm{inv} &\triangleq \begin{bmatrix} 1/m I_{3\times3} & 1/m I_{3\times3} & 1/m I_{3\times3} & 1/m I_{3\times3} \\ I_\mathrm{b}^{-1}[r_1]_\times & I_\mathrm{b}^{-1}[r_2]_\times & I_\mathrm{b}^{-1}[r_3]_\times & I_\mathrm{b}^{-1}[r_4]_\times \end{bmatrix} \\ g_b &\triangleq \begin{bmatrix} R_\mathrm{base}^\mathrm{world} \begin{bmatrix} 0 & 0 & -mg \end{bmatrix}^\top \\ 0 \end{bmatrix}. \end{aligned} \quad (37)$$

ACKNOWLEDGMENT

The authors would like to thank Bo Yang and Alapati Tuerxun for their help with the experiments.

REFERENCES

[1] C. D. Bellicoso et al., "Advances in real-world applications for legged robots," *J. Field Robot.*, vol. 35, no. 8, pp. 1311–1326, 2018.
[2] M. Tranzatto et al., "CERBERUS in the DARPA subterranean challenge," *Sci. Robot.*, vol. 7, no. 66, 2022, Art. no. eabp9742.
[3] P. Arm et al., "Scientific exploration of challenging planetary analog environments with a team of legged robots," *Sci. Robot.*, vol. 8, no. 80, 2023, Art. no. eade9548.
[4] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Trans. Robot.*, vol. 40, pp. 43–63, Oct. 2023.
[5] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 2245–2252.

[6] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3402–3421, Oct. 2023.

[7] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified MPC framework for whole-body dynamic locomotion and manipulation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4688–4695, Jul. 2021.

[8] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, "Whole-body MPC and online gait sequence generation for wheeled-legged robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2021, pp. 8388–8395.

[9] M. Bjelonic et al., "Offline motion libraries and online MPC for advanced mobility skills," *Int. J. Robot. Res.*, vol. 41, no. 9/10, pp. 903–924, 2022.

[10] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "TAMOLS: Terrain-aware motion optimization for legged systems," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3395–3413, Dec. 2022.

[11] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 558–564.

[12] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2001, vol. 1, pp. 239–246.

[13] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *Proc. Int. Conf. Robot. Autom*, 2019, pp. 8484–8490.

[14] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.

[15] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, 2020, Art. no. eabc5986.

[16] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2022, Art. no. eaau5872.

[17] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, 2022, Art. no. eabk2822.

[18] V. Makoviychuk et al., "Isaac Gym: High performance GPU based physics simulation for robot learning," in *Proc. 35th Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track (Round 2)*, 2021.

[19] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. Conf. Robot Learn.*, 2022, pp. 91–100.

[20] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Proc. Robot.: Sci. Syst.*, 2021.

[21] D. Kim, H. Kwon, J. Kim, G. Lee, and S. Oh, "Stage-wise reward shaping for acrobatic robots: A constrained multi-objective reinforcement learning approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2025, pp. 10268–10274.

[22] F. Shi, C. Zhang, T. Miki, J. Lee, M. Hutter, and S. Coros, "Rethinking robustness assessment: Adversarial attacks on learning-based quadrupedal locomotion controllers," in *Proc. Robot.: Sci. Syst.*, 2024.

[23] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *J. Math. Imag. Vis.*, vol. 40, pp. 120–145, 2011.

[24] Y. Lu, Z. Li, Y. Zhou, N. Li, and Y. Mo, "MPC-inspired reinforcement learning for verifiable model-free control," in *Proc. 6th Annu. Learn. Dyn. Control Conf.*, 2024, pp. 399–413.

[25] 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," *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6619–6626, Oct. 2023.

[26] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "DTC: Deep tracking control," *Sci. Robot.*, vol. 9, no. 86, 2024, Art. no. eadh5401.

[27] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 2908–2927, Oct. 2022.

[28] S. Yu, N. Perera, D. Marew, and D. Kim, "Learning generic and dynamic locomotion of humanoids across discrete terrains," in *Proc. IEEE-RAS 23rd Int. Conf. Humanoid Robots*, 2024, pp. 1048–1055.

[29] Z. Xie, X. Da, B. Babich, A. Garg, and M. v. de Panne, "Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," in *Proc. Int. Workshop Algorithmic Found. Robot*, 2022, pp. 523–539.

[30] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Real-time trajectory adaptation for quadrupedal locomotion using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom*, 2021, pp. 5973–5979.

[31] D. Hoeller, F. Farshidian, and M. Hutter, "Deep value model predictive control," in *Proc. Conf. Robot Learn.*, 2020, pp. 990–1004.

[32] A. Romero, Y. Song, and D. Scaramuzza, "Actor-critic model predictive control," in *Proc. IEEE Int. Conf. Robot. Autom*, 2024, pp. 14777–14784.

[33] R. Reiter, A. Ghezzi, K. BaumgÃĪrtner, J. Hoffmann, R. D. McAllister, and M. Diehl, "AC4MPC: Actor-critic reinforcement learning for nonlinear model predictive control," *IEEE Trans. Control Syst. Technol.*, vol. 34, no. 1, pp. 395–410, Jan. 2026.

[34] Y. Chen and Q. Nguyen, "Learning agile locomotion and adaptive behaviors via RL-augmented MPC," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 11436–11442.

[35] A. S. Morgan, D. Nandha, G. Chalvatzaki, C. D'Eramo, A. M. Dollar, and J. Peters, "Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 6672–6678.

[36] G. Grandesso, E. Alboni, G. P. R. Papini, P. M. Wensing, and A. Del Prete, "CACTO: Continuous actor-critic with trajectory optimization—Towards global optimality," *IEEE Robot. Autom. Lett.*, vol. 8, no. 6, pp. 3318–3325, Jun. 2023.

[37] E. Alboni, G. Grandesso, G. P. R. Papini, J. Carpentier, and A. Del Prete, "CACTO-SL: Using Sobolev learning to improve continuous actor-critic with trajectory optimization," in *Proc. 6th Annu. Learn. Dyn. Control Conf.*, 2024, pp. 1452–1463.

[38] J. Wu, G. Xin, C. Qi, and Y. Xue, "Learning robust and agile legged locomotion using adversarial motion priors," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 4975–4982, Aug. 2023.

[39] Y. Fuchioka, Z. Xie, and M. Van de Panne, "OPT-Mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 5092–5098.

[40] P. Gupta, J. M. Smereka, and Y. Jia, "Reinforcement learning compensated model predictive control for off-road driving on unknown deformable terrain," 2024, *arXiv:2408.09253*.

[41] Y. Yang et al., "CaJun: Continuous adaptive jumping using a learned centroidal controller," in *Proc. Conf. Robot Learn.*, 2023, pp. 2791–2806.

[42] H. J. Lee, S. Hong, and S. Kim, "Integrating model-based footstep planning with model-free reinforcement learning for dynamic legged locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 11248–11255.

[43] X. Da et al., "Learning a contact-adaptive controller for robust, efficient legged locomotion," in *Proc. Conf. Robot Learn*, PMLR, 2021, pp. 883–894.

[44] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Proc. Conf. Robot Learn.*, 2022, pp. 22–31.

[45] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.

[46] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *J. Mach. Learn. Res.*, vol. 18, no. 167, pp. 1–51, 2018.

[47] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 22–31.

[48] J. Lee, L. Schroth, V. Klemm, M. Bjelonic, A. Reske, and M. Hutter, "Exploring constrained reinforcement learning algorithms for quadrupedal locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 11132–11138.

[49] Y. Ma, A. Cramariuc, F. Farshidian, and M. Hutter, "Learning coordinated badminton skills for legged manipulators," *Sci. Robot.*, vol. 10, no. 102, 2025, Art. no. eadu3922.

[50] Y. Liu, J. Ding, and X. Liu, "IPO: Interior-point policy optimization under constraints," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 4940–4947.

[51] Y. Kim et al., "Not only rewards but also constraints: Applications on legged robot locomotion," *IEEE Trans. Robot.*, vol. 40, pp. 2984–3003, May 2024.

[52] J. Kim, J. Lee, and A. D. Ames, "Safety-critical coordination of legged robots via layered controllers and forward reachable set based control barrier functions," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 3478–3484.

[53] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," in *Proc. Robot.: Sci. Syst.*, 2024.

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

[55] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3642–3649, Apr. 2020.
[56] E. Chane-Sane, P.-A. Leziart, T. Flayols, O. Stasse, P. Souères, and N. Mansard, "CaT: Constraints as terminations for legged locomotion reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 13303–13310.
[57] E. Chane-Sane, J. Amigo, T. Flayols, L. Righetti, and N. Mansard, "Soloparkour: Constrained reinforcement learning for visual locomotion from privileged experience," in *Proc. Conf. Robot Learn.*, 2024, pp. 4268–4285.
[58] S. Gu et al., "A review of safe reinforcement learning: Methods, theories, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 11216–11235, Dec. 2024.
[59] M. Schubiger, G. Banjac, and J. Lygeros, "GPU acceleration of ADMM for large-scale quadratic programming," *J. Parallel Distrib. Comput.*, vol. 144, pp. 55–67, 2020.
[60] S. H. Jeon, S. Hong, H. J. Lee, C. Khazoom, and S. Kim, "CusADi: A GPU parallelization framework for symbolic expressions and optimal control," *IEEE Robot. Autom. Lett.*, vol. 10, no. 2, pp. 899–906, Feb. 2025.
[61] D. Cole, S. Shin, F. Pacaud, V. M. Zavala, and M. Anitescu, "Exploiting GPU/SIMD architectures for solving linear-quadratic MPC problems," in *Proc. Amer. Control Conf.*, 2023, pp. 3995–4000.
[62] A. Du, E. Adabag, G. Bravo, and B. Plancher, "GATO: GPU-accelerated and batched trajectory optimization for scalable edge model predictive control," 2025, *arXiv:2510.07625*.
[63] L. Amatucci, J. Sousa-Pinto, G. Turrisi, D. Orban, V. Barasuol, and C. Semini, "Primal-dual iLQR for GPU-accelerated learning and control in legged robots," *IEEE Robot. Autom. Lett.*, vol. 11, no. 1, pp. 1010–1017, Jan. 2026.
[64] A. L. Bishop, J. Z. Zhang, S. Gurumurthy, K. Tracy, and Z. Manchester, "RELU-QP: A GPU-accelerated quadratic programming solver for model-predictive control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 13285–13292.
[65] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 136–145.
[66] E. K. Ryu and W. Yin, *Large-Scale Convex Optimization: Algorithms and Analyses via Monotone Operators*. Cambridge, U.K.: Cambridge Univ. Press, 2022.
[67] Y. Chen, D. Tse, P. Nobel, P. Goulart, and S. Boyd, "CuClarabel: GPU acceleration for a conic optimization solver," 2024, *arXiv:2412.19027*.
[68] M. Blondel et al., "Efficient and modular implicit differentiation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 5230–5242.
[69] K. Tracy and Z. Manchester, "On the differentiability of the primal-dual interior-point method," 2024, *arXiv:2406.11749*.
[70] oswinso, "jaxproxqp," 2024. [Online]. Available: http://github.com/oswinso/jaxproxqp
[71] D. Applegate, M. Dıáz, H. Lu, and M. Lubin, "Infeasibility detection with primal-dual hybrid gradient for large-scale linear programming," *SIAM J. Optim.*, vol. 34, no. 1, pp. 459–484, 2024.
[72] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT Press, 1986.
[73] Q. Le Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier, "Contact models in robotics: A comparative analysis," *IEEE Trans. Robot.*, vol. 40, pp. 3716–3733, Jul. 2024.
[74] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[75] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.
[76] Unitree Robotics, "Unitree Go1," 2024. Accessed: Dec. 4, 2024. [Online]. Available: https://www.unitree.com/go1

Xu Yang received the B.Sc. degree in automation from the Department of Automation, Tsinghua University, Beijing, China, in 2021, where he is currently working toward the Ph.D. degree in automation with the Department of Automation, under the supervision of Prof. Yilin Mo.

His research interests involve legged locomotion and teleoperation.

Run Wang received the B.Sc. degree in automation from the Department of Automation, Tsinghua University, Beijing, China, in 2024, where he is currently working toward the Ph.D. degree in automation with the Department of Automation, under the supervision of Prof. Yilin Mo.

His research interests include reinforcement learning and optimal control in legged locomotion.

Yiwen Lu received the Bachelor of Engineering and Ph.D. degrees in automation from the Department of Automation, Tsinghua University, Beijing, China, in 2020 and 2025, respectively.

His research interests include adaptive and learning-based control, with applications in robotics.

Yilin Mo (Member, IEEE) received the Bachelor of Engineering degree in automation from the Department of Automation, Tsinghua University, Beijing, China, in 2007, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2012.

He is currently an Associate Professor with the Department of Automation, Tsinghua University. His research interests include reinforcement learning, optimal control, teleoperation, secure control systems, and networked control systems, with applications in sensor networks, power grids and robotics.