

Supplementary Materials for: Learning Quadruped Locomotion Using Differentiable Simulation

Anonymous Author(s)

Affiliation

Address

email

1 Double Integrator

A double integrator system is characterized by its position and velocity. The control input u directly controls the acceleration of the system. For a fair comparison, we formulate the problem as a discrete-time finite-horizon optimal control problem. The discrete-time state space representation of a double integrator is

$$x_{k+1} = Ax_k + Bu_k$$

where $A = \begin{bmatrix} 1 & d_t \\ 0 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} d_t^2/2 \\ d_t \end{bmatrix}$. Here, d_t is the simulation time step. The objective is to minimize a quadratic loss function

$$J = x_N^T Q_f x_N + \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k)$$
$$Q = Q_f = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = [1].$$

This is a discrete-time finite-horizon Linear-Quadratic Regulator problem, where the optimal control law can be found using dynamic programming

$$u_k^* = -(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A x_k, \quad k = 0, 1, \dots, N-1,$$

where $P_N = Q_f$ and P_k can be found from the Riccati recursion.

2 Experimental Setup

Simulation Setup: We develop our own differentiable simulation using PyTorch and CUDA. Our differentiable simulator allows for both forward propagation of the robot dynamics and backpropagation of the policy gradient. Additionally, we run IsaacGym alongside our differentiable simulation and use it to align the robot state resulting from our simplified robot dynamics. IsaacGym simulates the whole-body dynamics and complicated contacts between the robot and its environment. Both simulations are parallelized on GPU. We used a discretized simulation time step of 0.002s and a control frequency of 100 Hz.

Observation and Action: The policy observation includes random commands ($\mathbf{cmd}_{\text{rand}}$) for the reference velocity, sinusoidal and cosinusoidal representations of gait phases, the base velocity (\mathbf{v}_{WB}), the base orientation (\mathbf{q}_{WB}), the angular velocity ($\boldsymbol{\omega}_B$), motor position deviations from default ($\mathbf{q} - \mathbf{q}_{\text{default}}$), and a projected gravity vector ($\mathbf{g}_{\text{projected}}$). The policy action $\delta \mathbf{q}$ is the desired joint position offset from the default joint position.

Observation	Dimension	Action	Dimension
$\mathbf{cmd}_{\text{land}}$	3	$\delta \mathbf{q}$	12
$\sin(\text{gait phase})$	4		
$\cos(\text{gait phase})$	4		
\mathbf{v}_{WB}	3		
\mathbf{q}_{WB}	4		
$\boldsymbol{\omega}_B$	3		
$\mathbf{q} - \mathbf{q}_{\text{default}}$	12		
$\mathbf{g}_{\text{projected}}$	3		

Table 1: Policy observation and action for quadruped locomotion.

3 Foot Trajectory Planning

We use the foot position loss $\|\mathbf{p}_{\text{foot}} - \mathbf{p}_{\text{foot}}^{\text{ref}}\|^2$ to provide a learning signal for the swing legs. This loss term is critical for the swing leg since it contains information about the motor position. Following prior works [1, 2], the swing leg trajectory can be computed by fitting a quadratic polynomial over the lift-off $\mathbf{p}_{\text{foot}}^{\text{lift}}$, mid-air $\mathbf{p}_{\text{foot}}^{\text{air}}$, and landing position $\mathbf{p}_{\text{foot}}^{\text{land}}$ of each foot, where the lift-off position is the foot location at the beginning of the swing phase, the landing position $\mathbf{p}_{\text{foot}}^{\text{land}}$ is calculated using the Raibert Heuristics [3], which is expressed as the following function

$$\mathbf{p}_{\text{foot}}^{\text{land}} = \mathbf{p}_{\text{foot}}^{\text{hip}} + \mathbf{v}^{\text{CoM}} T_{\text{stance}} / 2$$

where T_{stance} is the expected time the foot will spend on the ground, $\mathbf{p}_{\text{foot}}^{\text{hip}}$ is the location on the ground beneath the robot’s hip, \mathbf{v}^{CoM} is the body velocity projected on the xy -plane. The desired contact state of each leg, e.g., swing or stance, is determined via a gait generator. The gait is modulated by a phase variable $\phi \in [0, 2\pi]$. The phase is defined through a dynamic function $\phi_{t+1} = \phi_t + 2\pi f \Delta t$, where f is the stepping frequency. We can design different locomotion patterns by adapting the stepping frequency f and the phase difference between each leg.

4 On the Importance of Non-differentiable Terminal Penalty

We highlight one important benefit of RL compared to differentiable simulation: RL can significantly enhance its robustness by directly optimizing through non-differentiable rewards or penalties. Specifically, we use a non-differentiable value $p = 200$ to penalize the robot when the robot experiences termination during training, e.g., falling on the ground or lifting its legs above its body.

$$r(\mathbf{x}_t, \mathbf{u}_t) = \begin{cases} -l(\mathbf{x}_t, \mathbf{u}_t) - p & \text{if termination} \\ -l(\mathbf{x}_t, \mathbf{u}_t) & \text{otherwise.} \end{cases}$$

Fig. 1 shows a study of using non-differentiable terminal penalty for both RL and differentiable simulation. The results show that adding a final penalty can greatly affect how well RL works. Without a penalty, RL might get trapped in a local minimum. However, with a large penalty at the end, RL can achieve better task rewards as well as more robust control performance. This is because RL optimizes a discounted return, which estimates “how good” it is to be in a given state. RL uses a state-value function to encode this information

$$V_{\pi}(s) = \mathbb{E}[G|S_0 = s] = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s \right].$$

On the other hand, a terminal penalty has no impact on differentiable simulation since the gradient of a constant value is equal to zero, and we do not leverage a state value function. As a result, differentiable simulation requires well-defined continuous functions, e.g., a potential function or control barrier functions for robust control.

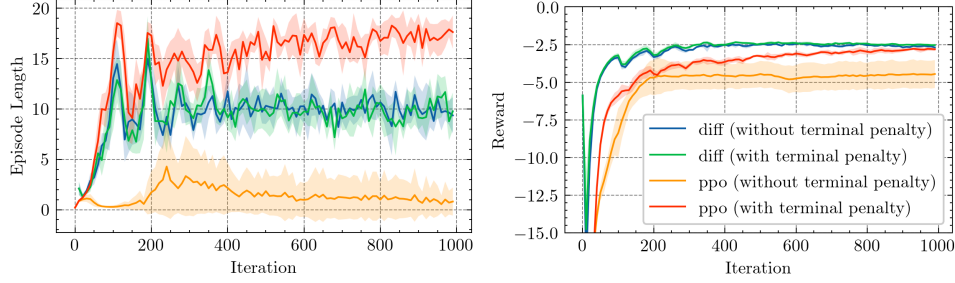


Figure 1: **A comparison of non-differentiable terminal penalty for policy training.** Using a non-differentiable terminal penalty, PPO can achieve robust control performance, e.g., longer episode length. We use 1024 robots for simulation.

5 Ablation Study

We conducted an ablation study to investigate the significance of the proposed state alignment mechanism. In our implementation, we use the state from IsaacGym to align the robot state within our simplified rigid-body differentiable simulation. To assess the impact of state alignment, we performed experiments where we removed the state alignment and compared the resulting learning curves to those obtained with state alignment. The results are presented in Figure 2. The results indicate that without state alignment, the robot fails to learn any useful walking skills.

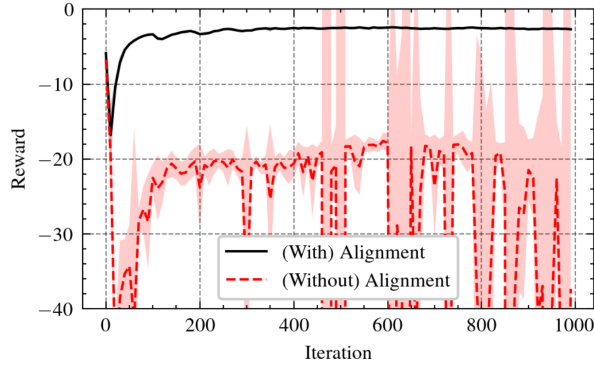


Figure 2: Ablation study for state alignment using IsaacGym.

6 Hyperparameters

Parameter	Value
learning rate	0.001
discount factor γ	0.95
GAE- λ	0.95
learning epoch	10
policy network	MLP [256, 256]
value network	MLP [256, 256]
clip range	0.2
entropy coefficient	0.002
number of epoch	10

Table 2: PPO hyperparameters.

Parameter	Value
learning rate	0.001
policy network	MLP [256, 256]
gradient decay factor α	0.9

Table 3: Hyperparameters for policy training using differentiable simulation.

7 Related Work

We provide more details about related work in legged locomotion.

Model-free Reinforcement Learning: Deep reinforcement learning has emerged as a dominant approach for developing control policies in legged locomotion [4, 5, 6, 7]. Recently, an important advancement in this area has been the introduction of IsaacGym [8], which is a GPU-accelerated simulator. IsaacGym can dramatically reduce the time required for dynamics simulation and policy training, enabling robots to learn to walk on flat ground in minutes [9]. This advancement has significantly accelerated the pace of research in legged locomotion, opening new avenues to address increasingly complex challenges [2, 10, 11, 12, 13]. Despite demonstrating strong performance, RL encounters challenges in scenarios where data generalization is slow, particularly with high-dimensional inputs. This limitation becomes evident as researchers start tackling vision-based problems, which inherently involve high-dimensional data. A common strategy involves a two-stage training process [13, 14, 6, 15, 7, 10]. First, a state-based RL policy—referred to as the *teacher*—is trained without the use of image data. Subsequently, this policy is used to guide the training of a *student* policy, which incorporates images as inputs. The requirement of combining RL with imitation learning highlights some fundamental challenges in RL and the need to investigate these challenges.

Differentiable Simulation: Differentiable simulation has recently gained momentum thanks to the development of differentiable physics simulators [16, 17, 18] and flexible automatic differentiation tools like Jax, DiffTaichi, Pytorch [19, 20, 21]. In principle, policy training through differentiable simulation allows for better convergence by replacing the zeroth-order gradient estimation of a stochastic objective with a more accurate estimate based on first-order gradients [22]. However, challenges such as the noisy optimization landscape and issues of exploding or vanishing gradients in long-horizon tasks render first-order gradient methods less effective. Several approaches have been developed to tackle these challenges. For instance, SHAC [23] addresses the gradient issue by truncating trajectories into several smaller segments, which helps manage exploding or vanishing gradients. Another notable example is DiffMimic [24], which focuses on mimicking motion trajectories for physically simulated characters. They show that differentiable simulation is a promising direction for improving sample efficiency. However, the capability of differentiable simulation to substantially speed up policy learning in contact-rich scenarios with physical interactions, as well as their practical applicability to real-world situations, such as quadruped locomotion, remains a significant challenge in robotics.

References

- [1] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots. Fast and efficient locomotion via learned gait transitions. In *Conference on Robot Learning*, pages 773–783. PMLR, 2022.
- [2] Y. Yang, G. Shi, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots. Cajun: Continuous adaptive jumping using a learned centroidal controller. *arXiv preprint arXiv:2306.09557*, 2023.
- [3] M. H. Raibert. *Legged robots that balance*. MIT press, 1986.
- [4] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

- [5] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- [6] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.
- [7] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.
- [8] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.
- [9] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.
- [10] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, pages 403–415. PMLR, 2023.
- [11] Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023.
- [12] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.
- [13] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. In *Conference on Robot Learning (CoRL)*, 2023.
- [14] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. *arXiv preprint arXiv:2309.14341*, 2023.
- [15] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.
- [16] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax—a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.
- [17] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.
- [18] T. Howell, S. Le Cleac’h, J. Bruedigam, Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 2022.
- [19] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/google/jax>.
- [20] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. DiffTaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.
- [21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

- 141 [22] H. J. Suh, M. Simchowitz, K. Zhang, and R. Tedrake. Do differentiable simulators give better
142 policy gradients? In *International Conference on Machine Learning*, pages 20668–20696.
143 PMLR, 2022.
- 144 [23] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin. Accel-
145 erated policy learning with parallel differentiable simulation. In *International Conference on*
146 *Learning Representations*, 2021.
- 147 [24] J. Ren, C. Yu, S. Chen, X. Ma, L. Pan, and Z. Liu. Diffmimic: Efficient motion mimicking
148 with differentiable physics. 2023.