Learning Quadruped Locomotion Using Differentiable Simulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: This work explores the potential of using differentiable simulation for learning robot control. Differentiable simulation promises fast convergence and stable training by computing low-variance first-order gradients using the robot model. Still, so far, its usage for legged robots is limited to simulation. The main challenge lies in the complex optimization landscape of robotic tasks due to discontinuous dynamics. This work proposes a new differentiable simulation framework to overcome these challenges. The key idea involves decoupling the complex whole-body simulation, which exhibits discontinuities by definition due to contact, into two separate continuous domains. Subsequently, we align the robot state resulting from the simplified model with a more precise, non-differentiable simulator to maintain sufficient simulation accuracy. Our framework enables learning quadruped walking in simulation in minutes without parallelization. When augmented with GPU parallelization, our approach allows the quadruped robot to master diverse locomotion skills on challenging terrains in minutes. We demonstrate that differentiable simulation outperforms a reinforcement learning algorithm (PPO) by achieving significantly better sample efficiency while maintaining its effectiveness in handling large-scale environments. Our policy achieves robust locomotion performance in the real world zero-shot.

Keywords: Differentiable Simulation, Legged Locomotion

1 Introduction

Many grazing animals, such as giraffes, lambs, and calves, can learn to stand and walk within minutes to hours after birth. How can legged robots master walking in minutes with a few trials and errors? Recent progress in legged robot control has been largely driven by model-free reinforcement learning (RL) and GPU-accelerated massive simulation [1, 2, 3, 4, 5, 6, 7, 8]. Central to these advancements is the policy search algorithm [9, 10, 11, 12], where the gradient of a control policy can be calculated via $\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)\nabla_\theta \log p(\tau;\theta)]$; which is a *zeroth-order* estimate of the true gradient of J based on sampled trajectories following the policy π_θ. Here, $R(\tau)$ is the reward of a trajectory τ. Such zeroth-order gradient estimate provides a powerful and general framework for robot control since it can handle non-differentiable optimization objectives and discontinuous dynamics. However, the gradient has high variances. Consequently, several additional strategies are required for stable training, such as using a clipped surrogate objective [13] and increasing the number of samples.

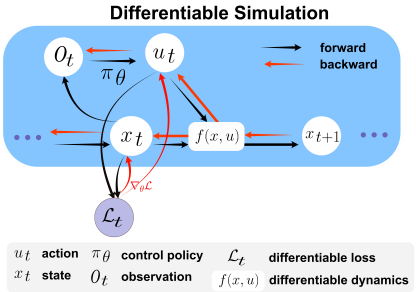

Figure 1: Graphical model for policy learning using differentiable simulation.

Submitted to the 8th Conference on Robot Learning (CoRL 2024). Do not distribute.

In robotics, leveraging well-established knowledges about robot dynamics can enable the construction of *first-order* gradient estimates $\nabla_\theta J(\theta) = \nabla_\theta R(\tau)$, which typically exhibit significantly lower variance than their zeroth-order counterparts and hold great potential for more stable training and faster convergence. Recently, policy training using first-order gradient has been notably advanced through *differentiable simulation [14, 15, 16, 17, 18]*; these works have shown promising results in reducing both the number of simulation samples and total training time compared to zeroth-order methods. Differentiable simulation combines the advantages of model-based control and data-driven learning. Like numerical optimal control, e.g., single shooting methods, differentiable simulation uses robot dynamics first to simulate the trajectory and then compute the gradients of the loss function with respect to the control parameters for optimization. Like RL, differentiable simulation can leverage large-scale simulation and deep learning to train a neural network policy that maps observations directly to control inputs. However, several challenges, such as discontinuous dynamics, complex optimization landscapes, and exploding or vanishing gradients in long-horizon tasks can render first-order gradient methods less effective.

This work explores the potential of policy training using differentiable simulation for quadruped locomotion. We show that differentiable simulation offers considerable advantages over model-free RL for policy training in legged locomotion. Notably, we demonstrate that a single robot, without the need for parallelization can quickly learn to walk within minutes in simulation using our approach. Leveraging the advantage of GPUs for parallelized simulation, our robot learns diverse walking skills over challenging terrain in minutes. Specifically, we train a quadruped robot to walk with different gait patterns, including trot, pace, bound, and gallop, and with varying gait frequencies. While model-free RL can also achieve similar performance given sufficient parallelization, our approach requires much less data, achieving significantly better sample efficiency while maintaining its effectiveness in handling large-scale environments. More importantly, we show that the policy trained via differentiable simulation can be transferred to the real world directly without fine-tuning. This work presents one of the earliest demonstrations of using differentiable simulation to control a real quadruped robot. This demonstration highlights the effectiveness of differentiable simulation for real-world applications in dynamic legged locomotion scenarios.

Contribution: The key to our approach is a novel policy training framework that combines the smooth gradients obtained from a simplified dynamics model for efficient backpropagation with the high fidelity of a more complex, non-differentiable simulator for accurate forward simulation. Instead of simulating the quadruped robot using whole-body dynamics, which by definition has discontinuities due to contact, we propose separating the simulation into its floating base space and its joint space. For the robot body, we employ an approximation using single rigid-body dynamics, which offers a continuous and effective representation of the robot body. To address simulation errors arising from our simplified rigid-body dynamics model, we incorporate a more precise, non-differentiable simulator. This non-differentiable simulator can simulate complex contact dynamics and is used to align the state in our simplified model, thereby ensuring that our training pipeline remains grounded in realistic dynamics. Figure 2 provides an overview of our system.

2 Methodology

2.1 Problem Formulation

We formulate legged robot control as an optimization problem. The robot is modeled as a discrete-time dynamical system, characterized by continuous state and control input spaces, denoted as $\mathcal{X}$ and $\mathcal{U}$, respectively. At each time step k, the system state is $x_k \in \mathcal{X}$, and the corresponding control input is $u_k \in \mathcal{U}$. An observation $o_k \in \mathcal{O}$ is generated at each time step based on the current state x_k through a sensor model $h : \mathcal{X} \to \mathcal{O}$, such that $o_k = h(x_k)$. The system's dynamics are governed by the function $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, which describes the time-discretized evolution of the system as $x_{k+1} = f(x_k, u_k)$. At each time step k, the robot receives a cost signal $l_k = l(x_k, u_k)$, which is a function of the current state x_k and the control input u_k. The control policy is represented as a deterministic, differentiable function such as a neural network $u_k = \pi_\theta(o_k)$. The neural network

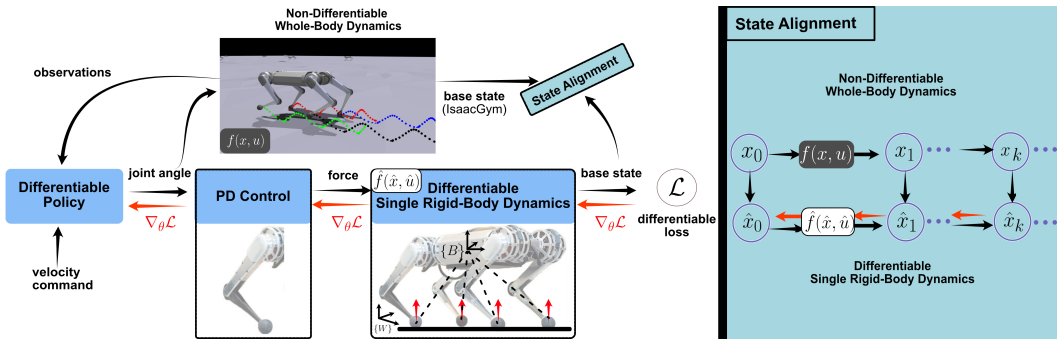

Figure 2: **System overview of learning quadruped locomotion using differentiable simulation.** Our approach decouples the robot dynamics into two separate spaces: joint and floating base spaces. We leverage the differentiability and smoothness of a single rigid-body dynamics for the robot's main body, which takes the ground reaction force from its legs as the control inputs. We use the state from a non-differentiable simulator (IsaacGym) to align the state in the differentiable simulation.

takes the observation o_k as input and outputs the control input u_k. The optimization objective is to find the optimal policy parameters θ^* by minimizing the total loss via gradient descent

$$\min_\theta \mathcal{L}_\theta = \sum_{k=0}^{N-1} l(x_k, u_k) = \sum_{k=0}^{N-1} l(x_k, \pi_\theta(o_k)) \tag{1}$$

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_\theta, \tag{2}$$

where α is the learning rate and $l(x_k, u_k)$ is the differentiable loss at simulation time step k.

2.2 Forward Simulation

Legged robots are characterized by complex dynamics resulting from their multi-body configuration and the contact between their body and the environment. The contacts introduce discontinuities in the dynamics, making first-order gradient-based optimization algorithms difficult. An important aspect of our approach involves separating the simulation of the robot's body from that of its joints. This design choice allows a simple implementation of the differentiable simulator.

First, we represent the robot's main body using single rigid-body dynamics. Single rigid-body dynamics have been shown to be useful for dynamic locomotion using online gradient-based optimization [19, 20]. The single rigid-body model offers a continuous representation of the robot base dynamics, avoiding the complex optimization landscape introduced by contacts. We develop our differentiable simulation using the single rigid-body dynamics, which is expressed as follows,

$$\dot{\mathbf{p}}_{WB} = \mathbf{v}_{WB} \qquad \dot{\mathbf{v}}_{WB} = \frac{1}{m}\sum_i \mathbf{f}_i + \mathbf{g}$$

$$\dot{\mathbf{q}}_{WB} = \frac{1}{2}\Lambda(\boldsymbol{\omega}_B)\cdot \mathbf{q}_{WB} \qquad \dot{\boldsymbol{\omega}}_B = \mathbf{I}^{-1}\left(\boldsymbol{\eta} - \boldsymbol{\omega}_B \times (\mathbf{I}\boldsymbol{\omega}_B)\right).$$

In this approximation, the state of our system is $\mathbf{x} = [\mathbf{p}, \mathbf{q}, \mathbf{v}, \boldsymbol{\omega}]$, where $\mathbf{p}_{\text{WB}} \in \mathbb{R}^3$ is the position and $\mathbf{v}_{\text{WB}} \in \mathbb{R}^3$ is the linear velocity of the center of mass in the world frame W, We use a unit quaternion $\mathbf{q}_{\text{WB}}$ to represent the orientation of the body in the world frame and use $\boldsymbol{\omega}_{\text{B}}$ to denote the body rates in the body frame B. Here, $\mathbf{I}$ is the robot's inertia tensor, $\boldsymbol{\eta}$ is the body torque, and $\boldsymbol{\Lambda}(\boldsymbol{\omega}_B)$ is a skew-symmetric matrix. The control inputs are the ground reaction force $\mathbf{f}_i$ from the legs that have contacts.

The ground reaction forces are required to simulate the rigid-body dynamics. One option for the control policy design is to output the ground reaction force directly, similar to the MPC design [19, 20]. Another option is to control the robot in the joint space, e.g., output desired joint position, which allows more control authority for the policy and adaptive behavior. In this case, the neural

network output (joint position) is required to be converted into the control input of the single rigid-body model, which is the ground reaction force. This conversion is achieved using a PD controller for forward propagation

$$\boldsymbol{\tau} = \mathbf{k_p}(\mathbf{q}^{\text{ref}} - \mathbf{q}) + \mathbf{k_d}(\dot{\mathbf{q}}^{\text{ref}} - \dot{\mathbf{q}}), \tag{3}$$

which calculates the required motor torque $\boldsymbol{\tau}$. Here, $\mathbf{k_p}$ and $\mathbf{k_d}$ are fixed gains, $\mathbf{q}^{\text{ref}}$ and $\mathbf{q}$ are the reference joint position and the current joint position separately, and $\dot{\mathbf{q}}^{\text{ref}}$ and $\dot{\mathbf{q}}$ are the reference joint velocity and the current joint velocity respectively. Subsequently, the motor torques are then converted to ground reaction forces $\mathbf{f}$ using the foot Jacobian $\mathbf{J}$: $\mathbf{f} = (\mathbf{J}^T)^{-1}\boldsymbol{\tau}$. The continuous nature of PD control enables the backpropagation of policy gradients. Consequently, our simulation framework treats the PD controller as a differentiable layer.

2.3 Backpropagation Through Time

In differentiable simulation for policy learning, the backward pass is crucial for computing the analytic gradient of the objective function with respect to the policy parameters. Following [21], the policy gradient can be expressed as follows

$$\nabla_\theta \mathcal{L}_\theta = \frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{i=1}^{k}\frac{\partial l_k}{\partial x_k}\prod_{j=i}^{k}\underbrace{\left(\frac{\partial x_j}{\partial x_{j-1}}\right)}_{\text{differentiable dynamics}}\frac{\partial x_i}{\partial \theta} + \frac{\partial l_k}{\partial u_k}\frac{\partial u_k}{\partial \theta}\right), \tag{4}$$

where the matrix of partial derivatives $\partial x_j / \partial x_{j-1}$ is the Jacobian of the dynamical system f. Therefore, we can compute the policy gradient directly by backpropagating through the differentiable physics model and a loss function l_k that is differentiable with respect to the system state and control inputs. A graphical model for gradient backpropagation in policy learning using differentiable simulation is given in Figure 1. Due to the usage of multiplication $\prod$, there are two potential issues in using Eq. (4) for policy gradient: 1) gradient vanishing or exploding, 2) long computation time. We tackle these two problems via short-horizon policy training.

2.4 State Alignment with A Non-Differentiable Simulator

Due to the simplification of our single rigid-body dynamics and the decoupling of the simulation space, the robot state can diverge from its actual one. Over time, even minor discrepancies can accumulate, leading to unrealistic states and ultimately causing the failure of policy training. We proposed to align the body state in our differentiable simulation using information from other simulators that use accurate whole-body dynamics. Specifically, we align the robot state in our differentiable simulation with the state information from IsaacGym [22].

We use the following equation to align the robot state

$$\hat{\mathbf{x}}_{t+1}^{\text{diff}} = \mathbf{x}_{t+1}^{\text{non-diff}} + \alpha * (\mathbf{x}_{t+1}^{\text{diff}} - \mathbf{x}_{t+1}^{\text{diff, detach}}). \tag{5}$$

Here, $\mathbf{x}_{t+1}^{\text{diff, detach}}$ and $\mathbf{x}_{t+1}^{\text{diff}}$ represent the same robot state of our differentiable simulator at time step $t+1$; hence, they share the same value. The word *detach* indicates that $\hat{\mathbf{x}}_{t+1}^{\text{diff, detach}}$ is detached from the computational graph for automatic differentiation. Therefore, we can reset the robot state using the value from the non-differentiable simulation during forward simulation: $\mathbf{x}_{t+1}^{\text{diff}} = \mathbf{x}_{t+1}^{\text{non-diff}} + \alpha * \mathbf{0}$. During backpropagation, the gradient of the state at a given time t is computed as follows

$$\partial \hat{\mathbf{x}}_{t+1}^{\text{diff}} / \partial \mathbf{x}_t^{\text{diff}} = \mathbf{0} + \alpha * \partial \mathbf{x}_{t+1}^{\text{diff}} / \partial \mathbf{x}_t^{\text{diff}}. \tag{6}$$

Here, $0 < \alpha \leq 1$ is used to decay the gradient.

The non-differentiable simulator can simulate complex contact dynamics and is used to align our simplified model, thereby ensuring that our differentiable training pipeline remains grounded in realistic dynamics. At the same time, our approach benefits from the simplified differentiable simulator, which offers smooth gradients for backpropagation. Fig. 2 shows the computational graph of the forward propagation and backpropagation using state alignment.

2.5 Short-Horizon Policy Training

Although smoothed physical models (e.g., single rigid-body dynamics) improve the local optimization landscape, the complexity of the optimization problem escalates significantly in long-horizon problems involving extensive concatenation of simulation steps. The situation further deteriorates when the actions within each step are interconnected through a nonlinear and nonconvex neural network control policy. The complexity of the resulting optimization landscape can make gradient-based methods to become trapped in local optima quickly. Instead of directly solving a long-horizon policy training problem, following [16, 4], where long-horizon simulation tasks are truncated into short-horizon simulation, we utilize short-horizon policy learning to achieve stable gradient back-propagation over a short horizon.

2.6 Differentiable Loss Function

A fundamental difference between policy training using differentiable simulation and reinforcement learning is that the loss function has to be differentiable when using differentiable simulation. RL allows the direct optimization of non-differentiable rewards, such as binary rewards $0/1$. On the contrary, differentiable simulation requires a smooth differentiable cost function to provide learning signals for the desired control inputs. We formulate a differentiable loss function $l(\mathbf{x}_t, \mathbf{u}_t)$ tailored for velocity tracking, where the main objective is to follow a specified velocity command, denoted as $\mathbf{v}^{\text{ref}}$. Additionally, we maintain the robot's body height, represented by p_z. To enhance the robustness of this system, we incorporate several regularization terms: one to mitigate a large angular velocity, thus controlling the body rate $\boldsymbol{\omega}$; another to limit the output action $\mathbf{u}$, preventing large control actions; and a term aimed at stabilizing the robot's orientation using projected gravity vector $\mathbf{g}_{\text{proj}}$. The loss function is defined as

$$\begin{aligned} l(\mathbf{x}_t, \mathbf{u}_t) \quad = \quad & b\|\mathbf{v} - \mathbf{v}^{\text{ref}}\|^2 + a\|p_z - p_z^{\text{ref}}\| + c\|\boldsymbol{\omega}\|^2 \\ & + d\|\mathbf{u}\|^2 + f\|\mathbf{g}_{\text{proj}}\| + g\|\mathbf{p}_{\text{foot}} - \mathbf{p}_{\text{foot}}^{\text{ref}}\|^2, \end{aligned} \tag{7}$$

where $\mathbf{p}_{\text{foot}}$ is the foot position.

3 Experimental Results

3.1 A Toy Example: Control of A Double Integrator

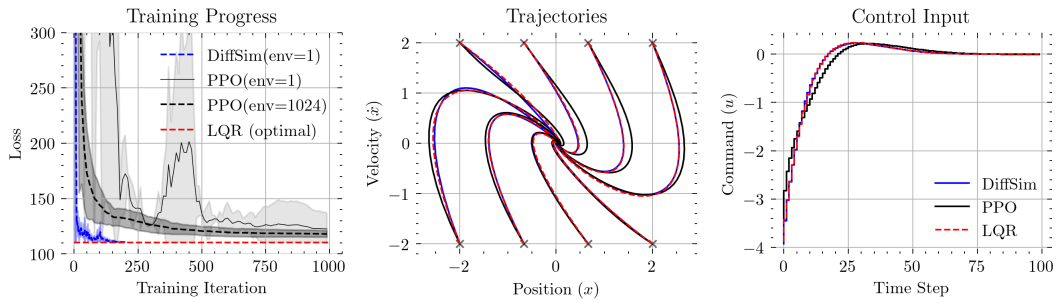

Figure 3: **Control of a double integrator using optimal control, reinforcement learning, and differentiable simulation.** (left): Learning curves. (middle): Trajectories of different control policies. We initialize the system at the same states for all methods. (right): DiffSim achieves control commands close to optimal control.

Inspired by [23], where the author studied the connection between reinforcement learning and optimal control using a toy double integrator example, we begin by examining the same toy example: control of a double integrator, which is a fundamental problem in system and control theory. The double integrator is a second-order control system that models simple point mass dynamics in one-dimensional space. The state variables are position x and velocity $\dot{x}$, with control inputs $u = \ddot{x}$. The

objective is to drive the system state from any initial point to the origin, $[x, \dot{x}] = [0, 0]$. Therefore, this problem can be solved by the Linear quadratic regulator (LQR), which can provide an optimal solution. We train a 2-layer multiplayer perception using backpropagation through time via differentiable simulation and the proximal policy optimization (PPO) method [13]. We show that the policy trained using differentiable simulation (DiffSim) achieves nearly optimal performance within a handful of training iterations and samples. In contrast, even when scaling the number of simulation environments to 1024 and training the policy with considerably more iterations, PPO fails to achieve the same level of control performance as LQR or DiffSim. This toy example suggests that scaling might be insufficient for PPO to achieve an optimal solution.

3.2 Learning to Walk with One Robot

This section explores whether a robot can learn to walk without parallelization and with very limited data points at each training iteration. We design a simple velocity-tracking task where the robot is required to follow a constant velocity in the x-axis, e.g., $v_x = 0.2\ m/s$. At each training iteration, we simulate 24-time steps with one single robot. Hence, each training iteration contains 24 data points. We trained the policy for a total of 1000 iterations. The result is given by the learning curve in Figure 4. Despite very limited data, our policy successfully learns to walk after minutes of training. As a comparison, PPO failed to achieve useful locomotion skills under the same condition. By providing first-order gradients with low variance, differentiable simulations offer a fundamental advantage: they allow for more efficient and reliable optimization of the control policy. This is because the gradient provides a clear direction for updating parameters, and the low variance ensures that this direction is consistent and reliable, even when data is limited.

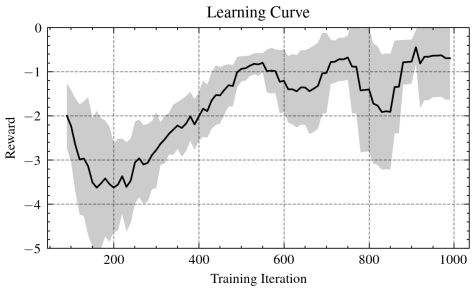

Figure 4: **Learning to walk with one simulated robot.** We run 10 experiments with different random seeds. The plot is smoothed using a moving average.

3.3 Learning Diverse Walking Skills on Challenging Terrains

We design a more difficult task: learning diverse locomotion skills over challenging terrains. Specifically, we design four different gait patterns, including *trot*, *pace*, *bound*, and *gallop*. Additionally, we vary the gait frequency from $1\,\mathrm{Hz}$ to $4\,\mathrm{Hz}$. The robot receives a high-level velocity command and is required to track randomly commanded velocity. Figure 5 shows the learning curves using different numbers of robots for policy training. We compare the learning performance of our method with a model-free RL algorithm (PPO) [13]. Given minimal samples, e.g., only four robots, RL has a slow convergence speed and does not learn meaningful walking skills, e.g., the policy constantly falls after a few simulation steps. In contrast, differentiable simulation achieves much higher rewards and can acquire useful walking skills.

As the number of robots increases, the performance of both algorithms improves. Notably, the performance improvement for RL is much more significant than our approach. This indicates that the zeroth-order gradient estimates used by reinforcement learning are generally inaccurate and require many more samples to achieve stable training. On the contrary, the first-order gradient estimates used by our differentiable simulation can have very stable and accurate gradients, even given very limited simulation samples. Figure 5 demonstrates diverse walking skills over challenging terrains using a blind policy trained via differentiable simulation.

We demonstrate the performance of our policy in the real world using Mini Cheetah [24]. Mini Cheetah is a small and inexpensive, yet powerful and mechanically robust quadruped robot, intended to enable the rapid development of control systems for legged robots. The robot uses custom back-

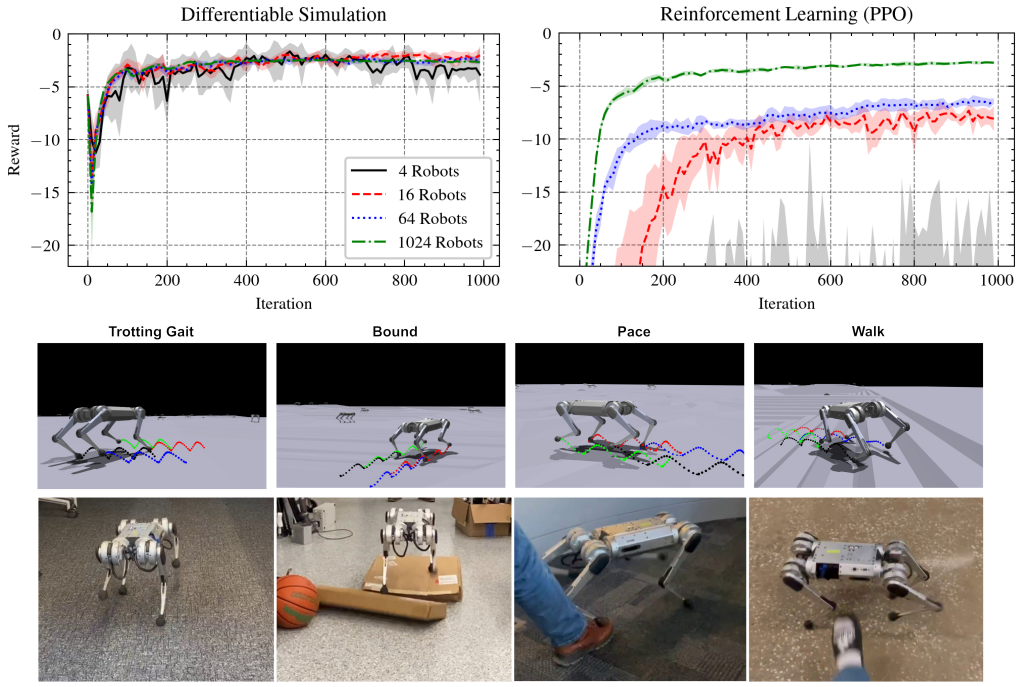

Figure 5: **Learning to walk on challenging terrains, reinforcement learning versus differentiable simulation.** Differentiable simulation exhibits significant advantages over PPO in terms of sample efficiency and learning stability. After training in simulation, the policy can be transferred to the real world without fine-tuning.

driveable modular actuators, which enable high-bandwidth force control, high force density, and robustness to impacts [24]. The control policy runs at 100 Hz during deployment. Figure 5 shows several snapshots of the robot's walking behavior over different terrains using different gait patterns. We trained a blind policy in simulation using 64 robots and then transferred the policy directly to the real world without fine-tuning. The robot can walk forward and backward with different gait patterns and frequencies. Moreover, the policy enables the robot to manage disturbances, such as unexpected forces and deformable terrain.

Additionally, we compare the total training wall-clock time and the final performance of the resulting policies. Differentiable simulation often involves the creation of a complex computational graph to facilitate gradient computation. As the simulation horizon extends, the length of the computational graph can increase proportionally, leading to a substantial increase in the total training time. This increase can diminish its benefits in situations where gathering a large number of samples is both cheap and fast. Therefore, we ask the question: *Can differentiable simulation be effectively applied to large-scale settings, for example, thousands of environments?* In our comparison, we ran both PPO and our training framework for the same number of iterations. We compare the total training time, which includes the time for simulating dynamics and updating policies. As shown in Table 1, our results indicate that differentiable simulation can be as time-efficient as PPO in large-scale environments despite the requirement of backpropagation through time (BPTT)—a process known for its computational demands.

Robots	Final Reward		Total Training Time [min]	
	PPO	**Ours**	PPO	**Ours**
4	-25.82 ± 2.16	$\mathbf{-3.83 \pm 3.01}$	7.27 ± 0.12	7.28 ± 0.12
16	-8.07 ± 1.30	$\mathbf{-2.03 \pm 0.69}$	7.54 ± 0.06	7.50 ± 0.08
64	-6.42 ± 0.29	$\mathbf{-2.49 \pm 0.37}$	8.46 ± 0.16	8.23 ± 0.07
1024	-2.80 ± 0.13	$\mathbf{-2.69 \pm 0.06}$	11.44 ± 0.13	11.55 ± 0.09

Table 1: **A comparison of final reward and total training time.** We run both methods for the same number of training iterations and collect the same amount of samples.

Related Work

Policy training through differentiable simulation has a strong connection with numerical optimal control and model-based reinforcement learning.

Model Predictive Control (MPC) is an optimal-control-based method that relies on fast numerical optimization and an accurate dynamics model to control the robot. Many methods use a reduced robot model, such as rigid-body single dynamics [19, 20, 25] to trade off between model accuracy and computational speed. Researchers have also studied control using more complex formulations, including centroidal [26] and whole-body dynamics [27]. Despite their success, MPC lacks the flexibility in its optimization objective design and robustness when facing disturbances or unexpected scenarios. Empirical results [1, 2, 3, 8] suggest that MPC tends to underperform when compared to data-driven approaches such as model-free deep RL. Instead of optimizing for an optimal trajectory like MPC, the decision variables of using differentiable simulation can be neural networks. Thus, it benefits from large-scale simulation for solving complex tasks and handling high-dimension inputs.

Model-based Reinforcement Learning [28, 29, 30, 31, 12] aims to enhance the sample efficiency of policy training by learning the dynamics models of robots and, often, their environments and a reward model as well. This approach typically begins with the assumption that the model of the robot is unknown and approximates the robot dynamics through methods like Gaussian Processes or neural networks. A key challenge with model-based RL is the exploitation of errors in the learned models. While black-box models, like deep neural networks, can accurately represent complex dynamics with high fidelity, their behavior outside of the training data distribution remains unpredictable. One promising direction to address this challenge is to incorporate physics-based priors into model learning. Methods such as Deep Lagrangian Networks [32, 33] or physics-informed deep learning [34, 35] leverage these priors to improve model reliability and generalization.

4 Limitations

A key advantage of model-free RL lies in its ability to directly optimize task-level rewards [36], especially non-differentiable rewards. In contrast, differential simulation depends on the backpropagation of information from its objective function, making it challenging to explore novel solutions guided by task objectives. Consequently, our system requires specifying foot positions using the Raibert heuristic [37], as it cannot explore foot placement motion through velocity tracking loss alone. In addition, RL's flexibility allows for more robust performance enhancements, such as implementing simple termination penalties to discourage falling behaviors or employing constant survival rewards to encourage continued operation. Differentiable simulation can potentially benefit from temporal-difference learning (TD) [11] since TD learning allows for iterative learning of a value function by bootstrapping from the current estimate of the value function without requiring a model of the environment. TD learning is particularly useful for learning a value function that can estimate long-term rewards, which are often hard to optimize directly, especially when the reward function is sparse or not differentiable.

5 Conclusion

This work proposes a novel perspective on learning quadruped locomotion policies using differentiable simulation by integrating a simplified rigid-body model with a non-differentiable model for whole-body dynamics. The learned locomotion policy is updated in this setting by directly back-propagating the differentiable loss function through the simplified model. At the same time, the robot's state is aligned using the more precise whole-body dynamics model. This framework introduces the advantages of faster and more stable training convergence of differential simulation to legged robots and ensures the accurate simulation of complex contact forces. We demonstrated that the proposed framework allows training a policy to walk in minutes using only one simulation environment. Additionally, we demonstrated that differentiable simulation is effective in large-scale environments. Therefore, this approach is promising for scenarios where data generation is slow and for addressing more complex challenges. Our framework holds great promise for addressing high-sample complexity tasks like vision-based navigation.

References

[1] 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.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[3] 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.

[4] 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.

[5] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.

[6] 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.

[7] 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.

[8] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[9] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[10] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[11] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[12] M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[14] 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.

[15] Z. Huang, Y. Hu, T. Du, S. Zhou, H. Su, J. B. Tenenbaum, and C. Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.

[16] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2021.

[17] J. Ren, C. Yu, S. Chen, X. Ma, L. Pan, and Z. Liu. Diffmimic: Efficient motion mimicking with differentiable physics. 2023.

[18] J. Xu, S. Kim, T. Chen, A. R. Garcia, P. Agrawal, W. Matusik, and S. Sueda. Efficient tactile simulation with differentiability for robotic manipulation. In *6th Annual Conference on Robot Learning*, 2022.

[19] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9. IEEE, 2018.

[20] S. Hong, J.-H. Kim, and H.-W. Park. Real-time constrained nonlinear model predictive control on so (3) for dynamic legged locomotion. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3982–3989. IEEE, 2020.

[21] L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

[22] 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.

[23] B. Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019.

[24] B. Katz, J. D. Carlo, and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301, 2019. doi:10.1109/ICRA.2019.8793865.

[25] G. Bledt and S. Kim. Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6316–6323. IEEE, 2019.

[26] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter. Perceptive locomotion through nonlinear model-predictive control. *IEEE Transactions on Robotics*, 2023.

[27] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.

[28] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[29] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[30] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[31] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, pages 2226–2240. PMLR, 2023.

[32] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations (ICLR 2019)*, 2019.

[33] M. Lutter, J. Silberbauer, J. Watson, and J. Peters. Differentiable physics models for real-world offline model-based reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4163–4170. IEEE, 2021.

[34] N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, and K. Um. Physics-based deep learning. *arXiv preprint arXiv:2109.05237*, 2021.

[35] H. Ma, Y. Zhang, N. Thuerey, X. Hu, and O. J. Haidn. Physics-driven learning of the steady navier-stokes equations using deep convolutional neural networks. *arXiv preprint arXiv:2106.09301*, 2021.

[36] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82): eadg1462, 2023.

[37] M. H. Raibert. *Legged robots that balance*. MIT press, 1986.