# Full-Order Sampling-Based MPC for Torque-Level Locomotion Control via Diffusion-Style Annealing

Haoru Xue*, Chaoyi Pan*, Zeji Yi, Guannan Qu, and Guanya Shi

***Abstract*— Due to high dimensionality and non-convexity, real-time optimal control using full-order dynamics models for legged robots is challenging. Therefore, Nonlinear Model Predictive Control (NMPC) approaches are often limited to reduced-order models. Sampling-based MPC has shown potential in nonconvex even discontinuous problems, but often yields suboptimal solutions with high variance, which limits its applications in high-dimensional locomotion. This work introduces DIAL-MPC (Diffusion-Inspired Annealing for Legged MPC), a sampling-based MPC framework with a novel diffusion-style annealing process. Such an annealing process is supported by the theoretical landscape analysis of Model Predictive Path Integral Control (MPPI) and the connection between MPPI and single-step diffusion. Algorithmically, DIAL-MPC iteratively refines solutions online and achieves both global coverage and local convergence. In quadrupedal torque-level control tasks, DIAL-MPC reduces the tracking error of standard MPPI by 13.4 times and outperforms reinforcement learning (RL) policies by 50% in challenging climbing tasks *without any training*. In particular, DIAL-MPC enables precise real-world quadrupedal jumping with payload. To the best of our knowledge, DIAL-MPC is the first *training-free* method that optimizes over full-order quadruped dynamics in real-time.**

## I. INTRODUCTION

Legged robots have demonstrated great potential in navigating through complex environments thanks to their agility and mobility [1–6]. However, the online control of articulated legged systems remains challenging because of their high-dimensional, underactuated and contact-rich nature, leads to non-convex and non-smooth optimization landscapes.

Reinforcement learning (RL) has become a popular approach for learning control policies for legged robots, thanks to its ease of implementation and strong performance in contact-rich problems [7–13]. However, RL suffers from time-consuming training and tedious tuning, and the resulting policies heavily depend on the training setup, limiting their test-time generalization to unseen tasks and environments. In the meanwhile, NMPC [5, 6, 14] is often limited to reduced-order models due to the intractability of solving full-order problems involving contacts and nonlinear dynamics.

Sampling-based MPC [15–18] has been applied to various nonlinear and hybrid dynamical systems because of its flexibility in handling arbitrary dynamics and constraints, as well as parallelizability. Nonetheless, these algorithms are sensitive to hyperparameters in high-dimensional and non-convex optimization problems, particularly the sampling kernel, leading to high variance and suboptimal performance.

* These authors contributed equally to this work.
The authors are with Carnegie Mellon University, USA. {haorux, chaoyip, zejiy, gqu, guanyas}@andrew.cmu.edu.
Paper website: https://lecar-lab.github.io/dial-mpc/

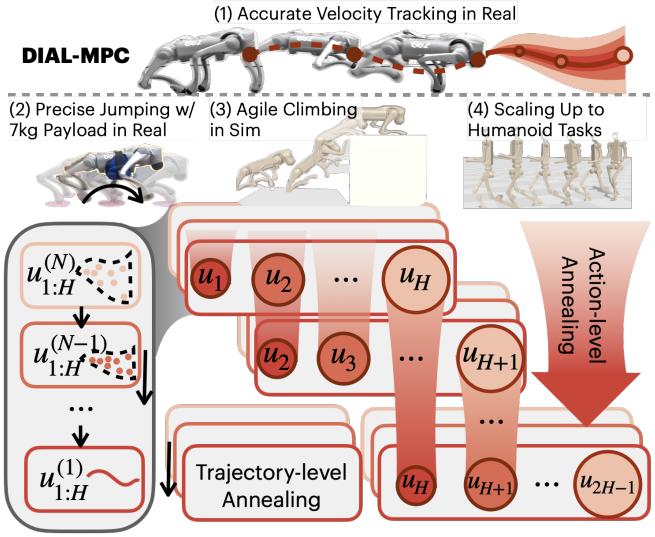


Fig. 1: Diffusion-inspired annealing for legged MPC (DIAL-MPC). To achieve both global coverage and local convergence, DIAL-MPC involves a bi-level diffusion-inspired annealing process. Trajectory-wise annealing is performed with different sampling variance. Action-wise annealing is performed on control input at different horizion. Over time, $u_H$ will be gradually refined by the two diffusion-inspired annealing processes, leading to a robust and efficient full-order online control.

Specifically, a large sampling range provides better global coverage but may result in solutions far from the optimum, while a small sampling range enhances local search ability but is more susceptible to local minima and initial guesses, leading to increased variance.

In this paper, we address these challenges by unveiling the intrinsic connection between sampling-based MPC and diffusion processes. Following the key iterative refinement idea of the diffusion model, we introduce a novel sampling-based MPC method, Diffusion-Inspired Annealing for Legged Model Predictive Control (DIAL-MPC). DIAL-MPC optimizes control sequences iteratively in a dual-loop manner. Specifically, DIAL-MPC starts optimizing the control sequence with smooth but inaccurate objectives and gradually shifts to more accurate local objectives.

Compared with MPPI, DIAL-MPC enables on-the-fly, full-order torque-level locomotion control by effectively balancing global coverage and local convergence. We evaluate DIAL-MPC on a quadrupedal robot with full-order dynamics and show that it can achieve real-time 50Hz control with both robustness and efficiency. As a training-free and purely

online method, DIAL-MPC enables precise jumping with payload and outperforms RL methods. The contribution of this paper is three-fold:

- **Novel Diffusion-Inspired Annealing Framework**: We propose a diffusion-inspired annealing framework for sampling-based MPC by revealing the connection between sampling-based MPC and diffusion processes.
- **Full-Order Torque-Level Control of Legged Robot**: We develop and implement DIAL-MPC for full-order torque-level control of legged robots based on the proposed annealing framework. To our knowledge, this is the first framework achieving both real-time flexibility and RL-level agility in legged locomotion.
- **Real-World Validation**: We validate the performance of DIAL-MPC for quadruped control, showing that it achieves real-time 50 Hz control with robustness and efficiency.

## II. RELATED WORK

### A. Agility in Legged Locomotion

Nonlinear Model Predictive Control (NMPC), particularly gradient-based methods, has demonstrated significant success in real-time control of legged locomotion with closed-loop stability [1, 6, 19, 20]. These approaches typically employ reduced-order models to mitigate the burden of planning over full-order hybrid dynamics, necessitating lower-level whole-body controllers for motion execution [2, 3, 5, 14].

However, reduced-order models can lead to sub-optimal performance and constraint violations, particularly under high agility demands. For instance, [4] introduces a tailored solver for full-order NMPC online, yet remains constrained by predefined contact sequences, limiting motion agility and robustness. In contrast, our method enables full-order model MPC without redundant constraints, allowing adaptation to real-time feedback and environmental interactions.

Model-free Reinforcement Learning (RL) has also been explored to address full-order control by learning optimal policies directly from high-fidelity simulations [11, 13, 21–24]. While these approaches eliminate the need for explicit modeling, they suffer from limited generalization to new tasks and dynamic environments.

Goal-conditioned RL [25, 26] enhances task-level generalization but still struggles with unseen dynamics and novel task classes. Our approach overcomes these limitations by enabling rapid motion generation through training-free online optimization, thereby combining the agility of RL with the robustness and generalization capabilities of MPC.

### B. Sampling-Based Optimization

Sampling-based or zeroth-order optimization methods, including Bayesian Optimization [27, 28], the Cross-Entropy Method [29], and Evolutionary Algorithms [30], are widely utilized for solving non-convex and non-smooth optimization problems. They differ from first-order methods as the gradient information is no longer required. These methods are particularly effective in applications such as hyperparameter tuning [31, 32] and generative modeling [33].

In the context of real-time robot control, Model Predictive Path Integral (MPPI) control [15] and its variants [16–18] have gained popularity for online motion planning [34–36], due to their inherent parallelizability and flexibility. Given a high-stiffness problem like legged locomotion, zeroth-order methods including policy gradient [37] have shown better convergence properties both empirically and theoretically [38] from their smoothing nature.

Despite their advantages, sampling-based methods are plagued by the curse of dimensionality and high variance, especially under constrained online sampling budgets. This is particularly problematic in high-dimensional, contact-rich environments.

### C. Parallel Robot Simulation

Massively parallelizable simulation environments, such as Isaac Gym [39], Brax [40], and MuJoCo [41], have become essential tools in the development of zeroth-order optimization methods for robot control. These simulators facilitate the rapid generation of data required by sample-hungry algorithms like PPO [42], enabling efficient training of complex policies.

## III. METHOD

In this section, we present our method by establishing the equivalence between MPPI and a single-stage diffusion process (section III-A). The connection then explains why the annealing process in diffusion helps MPPI to optimize over a non-smooth landscape, as discussed in section III-B. Leveraging this equivalence, we introduce a diffusion-inspired annealing technique for MPPI in section III-C.

### A. Sampling-Based MPC as Single-stage Diffusion

**Optimal control problem.** Sampling-based MPC aims to solve the following optimization problem:

$$\begin{aligned} \min_{u_{t:t+H}} \; & J(u_{t:t+H}) = \sum_{h=0}^{H} c(x_{t+h}, u_{t+h}) + c_f(x_{t+H+1}), \\ \text{s.t.} \quad & x_{t+h+1} = f(x_{t+h}, u_{t+h}) \quad \forall h \in \{0, \dots, H\}, \\ & x_{t:t+H+1} \in \mathcal{X}, \; u_{t:t+H} \in \mathcal{U} \end{aligned}$$

where $x_{t+h}$ is the state at time $t+h$, $u_{t+h}$ is the control input at time $t+h$, $f$ is the system dynamics, $c$ and $c_f$ are the cost function and terminal cost function, $\mathcal{X}$ and $\mathcal{U}$ are the state and control constraints, respectively.

MPPI estimates the optimal control sequence through the following steps: First, draw $N_W$ perturbations from a Gaussian distribution $[W]_i \sim \mathcal{N}(0, \Sigma_{t:t+H}), i = 1, \dots, N_W$ (which we collectively denote as $[W]_{1:N_W}$). Then the cost function $J(u_{t:t+H})$ is evaluated for each sampled control sequence by rolling out the system dynamics and cumulatively summing the cost. For each perturbed control sequence $U + [W]_i$, where $U = u_{t:t+H}$, evaluate the cost function $J(U + [W]_i)$ by simulating the system dynamics and accumulating the costs. Finally, update the control sequence

using temperature $\lambda$:

$$U^+ = U + \frac{\sum_{i=1}^{N_W} \exp\left(-\frac{J(U+[W]_i)}{\lambda}\right)[W]_i}{\sum_{j=1}^{N_W} \exp\left(-\frac{J(U+[W]_j)}{\lambda}\right)}, \tag{1}$$

**MPPI as a single-stage Diffusion.** The optimization problem can be reframed in a sampling context. Define the target distribution $p_0(U) \propto \exp\left(-\frac{J(U)}{\lambda}\right)$. As $\lambda \to 0$, samples from $p_0(U)$ concentrate around the optimal control sequence $U^*$ as illustrated in fig. 2. However, directly sampling from $p_0(U)$ is impractical due to its narrow support. To facilitate the sampling, we convolve $p_0(U)$ with a Gaussian noise kernel $\phi(\cdot)$, i.e., the density of $\mathcal{N}(0, \Sigma)$ to get the corrupted distribution $p_1(\cdot) \propto (p_0 * \phi)(\cdot)$ as depicted in fig. 4.

*Proposition 1 (Adopted from [43]):* The MPPI update (1) can be viewed as a one-step ascent with the score function $\nabla \log p_1(U)$ with a learning rate $\Sigma$:

$$U^+ = U + \Sigma \cdot \nabla \log p_1(U). \tag{2}$$

*Proof:* The diffused distribution $p_1$ is defined as $p_1(\cdot) \propto (p_0 * \phi)(\cdot)$ as in fig. 4. The score function $\nabla \log p_1(U)$ can be calculated in the following way:

$$\nabla \log p_1(U) = \frac{\nabla p_1(U)}{p_1(U)} = \frac{\nabla\left(p_0(U) * \phi(U)\right)}{p_1(U)} \tag{3a}$$

$$= \frac{p_0(U) * \nabla\phi(U)}{p_1(U)} = -\frac{p_0(U) * (\phi(U)\Sigma^{-1}U)}{p_1(U)} \tag{3b}$$

$$= -\Sigma^{-1}\frac{\int p_0(U-W)\phi(W)W dW}{\int p_0(U-W)\phi(W)dW} \tag{3c}$$

$$= -\Sigma^{-1}\frac{\mathbb{E}_{W\sim\phi(\cdot)}\left[p_0(U-W)W\right]}{\mathbb{E}_{W\sim\phi(\cdot)}\left[p_0(U-W)\right]} \tag{3d}$$

$$= \Sigma^{-1}\frac{\mathbb{E}_{W\sim\phi(\cdot)}\left[p_0(U+W)W\right]}{\mathbb{E}_{W\sim\phi(\cdot)}\left[p_0(U+W)\right]} \tag{3e}$$

$$\approx \Sigma^{-1}\frac{\sum_{i=1}^{N_W} \exp\left(-\frac{J(U+[W]_i)}{\lambda}\right)[W]_i}{\sum_{j=1}^{N_W} \exp\left(-\frac{J(U+[W]_j)}{\lambda}\right)}. \tag{3f}$$

From (3a) to (3b), we move the gradient into the convolusion. In (3b), the gradient of Gaussian kernal $\phi(W)$ is calculated. From (3b) to (3c), we use the definition of convolution. In (3d), we rewrite the integral as an expectation. In (3e), we flip the sign of $W$ to match the form of MPPI since the Gaussian distribution is symmetric. From (3e) to (3f), Monte Carlo approximation is applied. ■

In other words, MPPI performs a "denoising" step with the score function $\nabla \log p_1(U)$ conditioned on a *fixed* noise level $\mathcal{N}(0, \Sigma)$ [44]. The key difference is that the score-based diffusion model [44] iteratively refines samples using different noise levels, which motivates our annealing design[1].

### B. Diffusion-Inspired Annealing

Given that MPPI is a single-stage diffusion that moves particles toward the stationary point of $p_1(\cdot) = (p_0 * \phi)(\cdot)$

[1]There are some other differences between (2) and diffusion models: (2) does not have scaling factor or extra noise. See more discussion in [43].

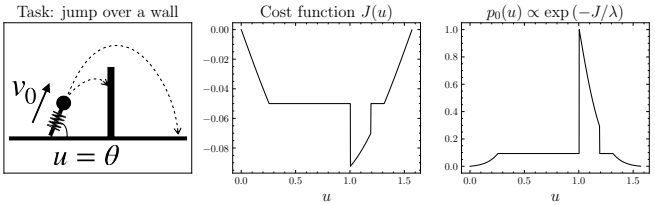


Fig. 2: Cost function $J(U)$ and target distribution $p_0(U)$ for a task where robot need to jump over a wall. The cost function could be highly non-convex and non-smooth due to the contact constraint. The resulting distribution $p_0(U)$ is also non-convex and sparse, which is hard to sample from.

(proposition 1), a natural question arises: What are the pros and cons of optimizing $p_1$ compared to directly solving for the optimum of $p_0$?

Figure 3 illustrates that the convexity of the distribution increases with progressively larger kernels. Each successive density function from $p_1$ to $p_4$ is obtained by convolving the original distribution $p_0$ with a progressively larger kernel. As discussed in section III-A, MPPI performs score ascent on $p_i$. Therefore, the primary **advantage** of conducting score ascent on $p_3$ and $p_4$ is that MPPI can more easily converge to the global optimum. The main **disadvantage** of optimizing a corrupted distribution is also clear from fig. 3. The optimum progressively shifts from $U^*$ to a distorted optimum under a larger kernel, thereby compromising the optimality of the original problem.

**Coverage and convergence trade-off.** The advantages and disadvantages highlight a fundamental trade-off between exploration and exploitation in MPPI: a larger $\det\Sigma$ promotes greater exploration (i.e., coverage) by widening the sampling distribution, which helps to avoid suboptimal local minima. However, a larger $\det\Sigma$ may compromise optimality(i.e., convergence) as it will introduce a larger optimality gap. In contrast, a smaller $\det\Sigma$ improves local optimality at the risk of getting trapped in local minima.

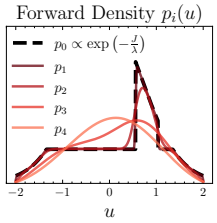


Fig. 3: Forward density function in diffusion process.

This trade-off becomes more pronounced in contact-rich tasks such as legged locomotion, where the optimization landscape is non-smooth, non-convex, and high-dimensional. The cost function $J$ and the distribution $p_0$ often have sharp and asymmetric peaks. In such cases, even a small increase in $\det\Sigma$ can degrade performance, and global optimality cannot be guaranteed through score ascent on $p_1(U)$ due to the small convolution kernel.

By fixing the sampling kernel size, MPPI may either over-explore or over-exploit, failing to balance exploration and convergence effectively. This concern is illustrated in fig. 4, which shows how different kernel sizes affect the performance of MPPI. Therefore, designing an effective sampling strategy that balances coverage and convergence

is crucial to unlocking the potential of MPPI in real-world legged locomotion tasks.

Fortunately, diffusion processes, known for their powerful sampling capabilities from complex distributions, offer a way to balance coverage and convergence through an annealing strategy. This strategy involves sampling iteratively at decreasing noise levels, effectively reversing the forward corruption process.

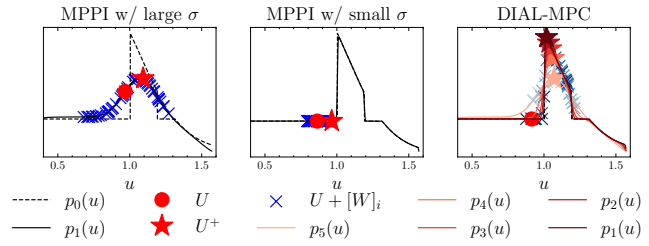


Fig. 4: Coverage and convergence trade-off in sampling-based methods. Given the target distribution $p_0(U)$ and the same number of samples, MPPI either over-explores or over-exploits the solution, while our method balances the exploration and exploitation to converge to optima following a diffusion-inspired annealing process.

**Annealing in diffusion process.** Instead of sampling solely from $p_1(\cdot)$, the forward process in diffusion defines a sequence of distributions with increasing noise levels: $p_1(\cdot), \ldots, p_{N-1}(\cdot), p_N(\cdot)$, where $N$ is the total number of diffusion stages. Each density is defined as $p_i(\cdot) = (p_0 * \phi_i)(\cdot)$ with $\phi_i(\cdot) \sim \mathcal{N}(0, \Sigma^i)$. Sampling proceeds in reverse order. Starting from a higher noise level $\Sigma^N$, the sampling distribution $p_N(\cdot)$ is highly spread out, ensuring good global exploration. As the noise level decreases, the sampling distributions $p_i(U)$ become more concentrated, refining the search towards the target distribution $p_0(U)$ and improving convergence to the optimal solution. We adopt an exponential schedule for the noise levels, inspired by diffusion processes, to design the sampling kernels:

$$\det\left(\Sigma^i\right) = \exp\left(-\frac{N-i}{\beta N} d\right), \quad \forall i \in \{N, \ldots, 1\}, \tag{4}$$

where $\beta$ is the temperature parameter for the annealing process, and $N$ is the number of iterations for the annealing process, $d$ is the dimension of the sampling space.

Given that MPPI corresponds to a single-stage diffusion process, we can naturally incorporate this multi-stage annealed diffusion approach to design an improved sampling strategy for MPPI. In section III-C, we discuss how to implement this diffusion-inspired annealing process within the MPPI framework in a receding horizon manner.

### C. Diffusion-Inspired Annealing for Sampling-Based MPC

Combining MPPI with multi-stage annealing, we propose DIAL-MPC (algorithm 1) that leverages the receding horizon structure of MPC and introduces a dual-loop covariance design. This design comprises two annealing procedures:

**Algorithm 1** Diffusion-Inspired Annealing for Legged MPC

Initialize $u_{0:H} \leftarrow \mathbf{0}$
**for** $t = 0$ to $T$ **do**
  **for** $i = 1$ to $N$ **do**
    Get diffusion noise kernel $\Sigma^i_{t:t+H}$ with (7).
    Sample $[W]_{1:N_W} \sim \mathcal{N}(0, \Sigma^i_{t:t+H})$.
    Rollout $u_{t:t+H} + [W]_{1:N_W}$ and evaluate the cost function $J(u_{t:t+H} + [W]_{1:N_W})$.
    Estimate Score $\nabla \log p_i(\cdot)$ with (3).
    Update $u^{(i)}_{t:t+H}$ with (2).
  **end for**
  Receding horizon $u_{t+1:t+H+1} \leftarrow \texttt{shift}(u_{t:t+H})$
**end for**

an outer-loop trajectory-level annealing and an inner-loop action-level annealing, which we detail below.

**Dual-loop annealing.** In a receding horizon MPC framework with a horizon length $H$ and $N$ update iterations for each control sequence at each timestep $t$, $u_H$ (the last element of $u_{0:H}$) is first updated at time 0 by $N$ times using convolution kernel $\Sigma^N_H \ldots, \Sigma^1_H$. After $u_0$ is applied to the system, the control sequence shifts forward. At the next time step $t = 1$, $u_H$ is updated again $N$ times, now with kernel $\Sigma^N_{H-1}, \ldots, \Sigma^1_{H-1}$ as $u_H$ becomes the second-to-last element of the updated control sequence $u_{1:H+1}$. This procedure continues, and by the time $u_H$ is applied to the system at $t = H$, it will have undergone $NH$ updates with convolution kernels: $\Sigma^N_H \ldots, \Sigma^1_H; \Sigma^N_{H-1}, \ldots, \Sigma^1_{H-1}; \ldots; \Sigma^N_0 \ldots, \Sigma^1_0$ as shown in the last graph of fig. 1. Essentially, each control action $u_h$ is updated in a dual-loop manner before being applied to the system. This motivates the design of two annealing procedures: the outer loop is a trajectory-level annealing schedule for all the control sequence at a certain stage $i$ (i.e. designing the overall size of $\Sigma^i_{t:t+H}$ for a given $i$) and the inner loop is an action-level annealing schedule for different control at different horizons (i.e. the size of $\Sigma^i_{t+h}$ for $h = 0, \ldots, H$).

**Trajectory-level annealing.** For the outer-loop trajectory-level annealing, we define the schedule as:

$$\det\left(\Sigma^i_{t:t+H}\right) \propto \exp\left(-\frac{N-i}{\beta_1 N} H d_u\right), \quad \forall i \in \{N, \ldots, 1\}, \tag{5}$$

where $\beta_1$ is the temperature parameter for the trajectory-level annealing, and $d_u$ is the dimension of a single control. The covariance matrix decreases over time as $i$ decreases, progressively narrowing the sampling distribution towards the target distribution $p_0$.

**Action-level annealing.** For the inner-loop action-level annealing, the schedule is given by:

$$\det\left(\Sigma^i_{t+h}\right) \propto \exp\left(-\frac{H-h}{\beta_2 H} d_u\right), \quad \forall h \in \{0, \ldots, H\}, \tag{6}$$

where $\beta_2$ is the temperature parameter for action-level annealing. The covariance matrix increases with time as $h$

increases, allowing for a larger sampling region for future control actions that have been updated fewer times compared to those at the front of the horizon.

Note that (5) and (6) specify only the overall size (i.e., the determinant) of the convolution kernels, leaving flexibility in designing the exact covariance matrices $\Sigma^i_{t+h}$. As a practical realization of (5) and (6), we assume that each $\Sigma^i_{t+h}$ is isotropic and define it as:

$$\Sigma^i_{t+h} = \exp\left(-\frac{N-i}{\beta_1 N} - \frac{H-h}{\beta_2 H}\right) I. \tag{7}$$

where $I$ is the identity matrix. This design combines both the outer-loop and inner-loop annealing schedules, adjusting the covariance matrices to ensure appropriate exploration and exploitation at each iteration and horizon step.

## IV. EXPERIMENT

In this section, we demonstrate the advantages of DIAL-MPC in terms of: (1) convergence and coverage, (2) efficiency in test-time generalizability, and (3) robustness to real-world model mismatch. We compare DIAL-MPC against MPPI and another sampling-based optimization method, CMA-ES [45]. In addition, we provide goal-conditioned reinforcement learning (GCRL) as a performance reference. Our results show that DIAL-MPC reduces the tracking error by 3.9 times in walking tasks compared to MPPI and outperforms GCRL on all tasks requiring both precision and agility, especially in the presence of significant model mismatch.

We design a set of agile locomotion tasks to showcase the performance of DIAL-MPC: (1) Walking Tracking: a quadruped robot[2] is tasked with tracking a desired linear velocity and a desired yaw rate, requiring precise control of the torso. (2) Sequential Jumping: a quadruped robot must jump onto a series of small circular platforms placed randomly, each with a radius of $10\,\mathrm{cm}$. (3) Crate Climbing: a quadruped robot is tasked with climbing a crate with a height of $60\,\mathrm{cm}$, which is more than twice the height of the robot. We leave the specific details of the tasks' implementation in Appendix B.

### A. Convergence and Coverage

To answer the question of whether DIAL-MPC is a better solver for the legged MPC problems, we compare the performance of DIAL-MPC with vanilla MPPI, CMA-ES and NMPC. Since vanilla MPPI only uses a single kernel size, we tested both MPPI with large kernel size $0.2$ (MPPI-explore) and small kernel $0.05$ (MPPI-exploit). Compared with DIAL-MPC, CMA-ES optimizes the sampling covariance in an evolutionary way. For all sampling-based MPC methods, we use the same number of samples $N_W = 2048$ and optimization steps $H = 20$ $(0.4\,\mathrm{s})$ to ensure a fair comparison. For NMPC, we use Mujoco MPC [36] as our baseline. As a reference, we also include the performance of GCRL, which is trained using PPO [42] over 500 million steps, which takes approximately 31 minutes. We share the same reward functions for all methods. Due to unstable exploration in GCRL, we added two additional reward terms to regularize the policy, whereas DIAL-MPC only requires six reward terms.

[2]Detailed information of the hardware setup can be found in A.

| | Walk Track ↓ | Seq Jump ↑ | Crate Climb ↑ |
|---|---|---|---|
| MPPI-explore | 0.190 | 0.440 | 0.3 |
| MPPI-exploit | 0.230 | 0.450 | 0.2 |
| CMA-ES | 0.544 | 0.345 | 0.2 |
| NMPC | 0.055 | - | - |
| GCRL* | 0.119 | 0.855 | 0.6 |
| **DIAL-MPC** | **0.024** | **0.885** | **0.9** |

TABLE I: Performance comparison of different methods over three tasks. For walk tracking, it represents the tracking error; For sequential jumping, it represents average total contact reward; For crate climbing, it represents success rate out of 10 trials of climbing onto random crates with heights ranging from 0.125 to 0.6 m. *GCRL requires offline training so it is not an apple-to-apple baseline.

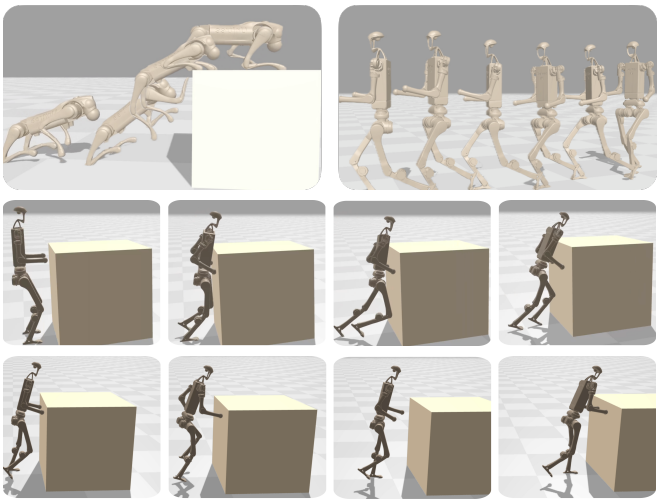

Fig. 5: **Top**: the coverage of DIAL-MPC in crate-climbing and humanoid jogging task. The crate-climbing task requires the robot to climb up a crate more than two times higher than itself. The full-size humanoid jogging task demands DIAL-MPC to handle a higher-dimensional action space of 19. **Middle**: DIAL-MPC controlling a humanoid pushing a $30\,\mathrm{kg}$ crate. **Bottom**: DIAL-MPC generates a motion strategy with less effort when reducing the crate's weight to $15\,\mathrm{kg}$.

**Convergence of DIAL-MPC.** As shown in table I, DIAL-MPC achieves the best performance across all tasks. Compared with other sampling-based MPC methods (namely MPPI and CMA-ES), DIAL-MPC generates better solutions, with $13.4$ times lower tracking error and $107.7\%$ higher contact reward, thanks to its diffusion-inspired annealing process. For the crate climbing task, DIAL-MPC is the only sampling-based MPC that consistently generates feasible solutions, improving the success rate by 3 times compared to the best MPPI scheduling. This highlights the superior coverage of the solution space by DIAL-MPC. Compared

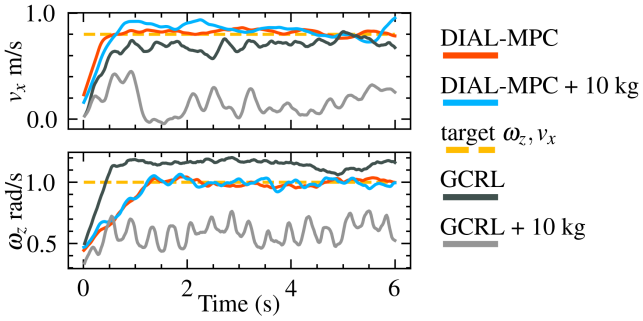


Fig. 6: The linear and angular velocity tracking performance of DIAL-MPC and GCRL in the walking task test in simulation. GCRL fails with a 10kg payload.

with NMPC, DIAL-MPC is able to solve tasks requiring higher agility and non-smooth costs, such as sequential jumping and crate climbing, where NMPC is more likely to get stuck in local minima and fail to converge.

The superior convergence of DIAL-MPC is further demonstrated when compared with GCRL, where DIAL-MPC consistently outperforms GCRL in all tasks even if we use a lower-level leg position controller for RL and DIAL-MPC directly outputs the torque. Although DIAL-MPC is training-free, making a direct comparison potentially unfair, these results showcase the advantage of the annealing process in generating finer solutions compared to the Gaussian exploration in RL.

**Coverage of DIAL-MPC.** DIAL-MPC is also capable of searching for non-trivial solutions and generating diverse motions. As examples, we visualize the solutions in the crate climbing task and a humanoid jogging task in Figure 5. DIAL-MPC generates diverse solutions in high-dimensional spaces, thanks to the annealing process.

### *B. Test-Time Generalizability*

As a training-free method, DIAL-MPC offers better test-time generalizability. Given a new task or model, DIAL-MPC can generate solutions in real time without finetuning.

**Task-level generalizability.** Compared with GCRL, DIAL-MPC reduces the tracking error by 3.9 times in the walking task and improves the contact reward by 3.5% given different goals. Figure 6 visualizes the tracking performance of both methods. DIAL-MPC achieves higher tracking accuracy thanks to its explicit conditioning on each task, whereas GCRL uses a universal policy for all tasks.

**Dynamic-level generalizability.** Another advantage of DIAL-MPC is its explicit conditioning on the dynamics model, enabling better adaptation to new dynamics given updated models. Figure 6 illustrates the tracking performance of both methods with a 10 kg payload attached to the robot's base. The crate-climbing task is deprecated due to the heavy payload leads to infeasible solutions. The GCRL policy is augmented with domain randomization of mass, actuator gain, and friction to improve robustness to model parameters. Table II shows the performance comparison of GCRL and DIAL-MPC under a 10 kg payload. Without explicit conditioning on the physical parameters, GCRL performs poorly in tracking and completely fails after adding the payload. In contrast, DIAL-MPC outperform GCRL with a larger margin in the jumping task, demonstrating the advantage of explicitly conditioning on the dynamics model. While RL can be enhanced by conditioning on physical parameters or history, this requires additional training time and engineering effort. Conversely, the training-free DIAL-MPC can be instantly deployed with an updated model.

| | Walk Track ↓ | Seq Jump ↑ |
|---|---|---|
| GCRL | 0.517 | 0.150 |
| **DIAL-MPC** | **0.036** | **0.815** |

TABLE II: Performance comparison of GCRL and DIAL-MPC under 10 kg payload.

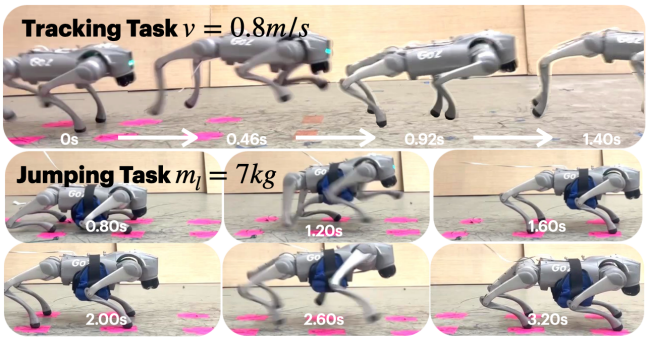


Fig. 7: Velocity tracking task and sequential jumping task with a 7 kg payload on Go2 quadruped in real world with DIAL-MPC for direct torque control. DIAL-MPC enables both precise and agile motion.

### *C. Robustness to Model Mismatch*

When dealing with unknown dynamics models, DIAL-MPC 's robustness stems from the noise injection process in the diffusion-inspired annealing. By controlling the final noise level, we can achieve a suboptimal but robust solution given a mismatched model. Since the sampling-based MPC baselines fail to operate effectively in the real world, we compare the performance of DIAL-MPC with the baseline in simulation with mismatched mass parameters. Table III shows the performance comparison of different methods under 2 kg base mass mismatch in simulation, where DIAL-MPC still outperforms the best sampling-based MPC baselines by 92% in the walking task and 126% in the jumping.

Figure 7 shows the DIAL-MPC control a Unitree Go2 robot to walk and jump. The robot is able to walk smoothly and jump precisely with direct torque control.

## V. CONCLUSION

This work presents DIAL-MPC, a sampling-based MPC method with a diffusion-inspired annealing process to balance coverage and convergence in real-world legged locomotion. DIAL-MPC can solve the full-order control problem

| | Walk Track ↓ | Seq Jump ↑ |
|---|---|---|
| MPPI-explore | 0.361 | 0.270 |
| MPPI-exploit | 0.407 | 0.200 |
| CMA-ES | 0.556 | 0.100 |
| **DIAL-MPC** | **0.188** | **0.610** |
| DIAL-MPC in real | 0.322 | 0.600 |

TABLE III: Performance comparison of different methods under 2 kg base mass mismatch in simulation and real world.

efficiently with the help of diffusion process and is generalizable to various tasks and dynamics in a training-free manner. One limitation is that DIAL-MPC requires fast simulation to generate samples, which limits the application of DIAL-MPC in longer planning horizon tasks. In the future, we plan to further accelerate and improve the sample efficiency by learning a nominal policy, value function and model as what has been done in model-base RL.

## REFERENCES


[1] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the MIT Cheetah 2: Control design and experiments," *Other repository*, Feb. 2017.

[2] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, *Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control*, Sep. 2019. arXiv: 1909.06586 [cs].

[3] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online Walking Motion Generation with Automatic Footstep Placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, Jan. 2010.

[4] C. Khazoom, S. Hong, M. Chignoli, E. Stanger-Jones, and S. Kim, *Tailoring Solution Accuracy for Fast Whole-body Model Predictive Control of Legged Robots*, Jul. 2024. arXiv: 2407.10789 [cs].

[5] J. Koenemann *et al.*, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 3346–3351.

[6] M. Neunert *et al.*, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018. arXiv: 1712.02889 [cs].

[7] I. Radosavovic *et al.*, *Humanoid Locomotion as Next Token Prediction*, Feb. 2024. arXiv: 2402.19469 [cs].

[8] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion,"

[9] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme Parkour with Legged Robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan: IEEE, May 2024, pp. 11 443–11 450.

[10] 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*, vol. 7, no. 62, eabk2822, Jan. 2022. arXiv: 2201.08117 [cs].

[11] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning,"

[12] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, *Learning Torque Control for Quadrupedal Locomotion*, Mar. 2023. arXiv: 2203.05194 [cs, eess].

[13] Z. Fu, A. Kumar, J. Malik, and D. Pathak, *Minimizing Energy Consumption Leads to the Emergence of Gaits in Legged Robots*, Oct. 2021. arXiv: 2111.01674 [cs].

[14] S. Kuindersma, F. Permenter, and R. Tedrake, "An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2589–2594. arXiv: 1311.1839 [cs].

[15] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1433–1440.

[16] Z. Yi, C. Pan, G. He, G. Qu, and G. Shi, *CoVO-MPC: Theoretical Analysis of Sampling-based MPC and Optimal Covariance Design*, Jan. 2024. arXiv: 2401.07369 [cs].

[17] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory Distribution Control for Model Predictive Path Integral Control using Covariance Steering," in *2022 International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, USA: IEEE, May 2022, pp. 1478–1484.

[18] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. Rehg, and E. Theodorou, "Robust Sampling Based Model Predictive Control with Sparse Objective Information," in *Robotics: Science and Systems XIV*, Robotics: Science and Systems Foundation, Jun. 2018.

[19] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, *Perceptive Locomotion through Nonlinear Model Predictive Control*, Aug. 2022. arXiv: 2208.08373 [cs].

[20] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, *A Unified MPC Framework for Whole-Body Dynamic Locomotion and Manipulation*, Mar. 2021. arXiv: 2103.00946 [cs].

[21] T. He *et al.*, "OmniH2O: Universal and Dexterous Human-to-Humanoid Whole-Body Teleoperation and Learning,"

[22] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "HumanPlus: Humanoid Shadowing and Imitation from Humans,"

[23] C. Zhang, W. Xiao, T. He, and G. Shi, "WoCoCo: Learning Whole-Body Humanoid Control with Sequential Contacts,"

[24] T. He *et al.*, *Learning Human-to-Humanoid Real-Time Whole-Body Teleoperation*, Mar. 2024. arXiv: 2403.04436 [cs, eess].

[25] D. Ghosh, A. Gupta, and S. Levine, *Learning Actionable Representations with Goal-Conditioned Policies*, Jan. 2019. arXiv: 1811.07819 [cs, stat].

[26] V. Atanassov, J. Ding, J. Kober, I. Havoutis, and C. Della Santina, *Curriculum-Based Reinforcement Learning for Quadrupedal Jumping: A Reference-free Design*, Mar. 2024. arXiv: 2401.16337 [cs].

[27] P. I. Frazier, *A Tutorial on Bayesian Optimization*, Jul. 2018. arXiv: 1807.02811 [cs, math, stat].

[28] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*, Nov. 2015. arXiv: 1503.03585 [cond-mat, q-bio, stat].

[29] S. Mannor, R. Rubinstein, and Y. Gat, "The Cross Entropy Method for Fast Policy Search,"

[30] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, "Natural Evolution Strategies," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, Hong Kong, China: IEEE, Jun. 2008, pp. 3381–3387.

[31] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012.

[32] J. M. Hernández-Lobato, J. Requeima, E. O. Pyzer-Knapp, and A. Aspuru-Guzik, "Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space," in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Jul. 2017, pp. 1470–1479.

[33] J. Song, C. Meng, and S. Ermon, "Denoising Diffusion Implicit Models," in *International Conference on Learning Representations*, Oct. 2020.

[34] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou, "$\mathcal{L}1$-Adaptive MPPI Architecture for Robust and Agile Control of Multirotors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 7661–7666.

[35] J. Sacks, R. Rana, K. Huang, A. Spitzer, G. Shi, and B. Boots, *Deep Model Predictive Optimization*, Oct. 2023. arXiv: 2310.04590 [cs].

[36] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, *Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo*, Dec. 2022. arXiv: 2212.00541 [cs, eess].

[37] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," in *Advances in Neural Information Processing Systems*, vol. 12, MIT Press, 1999.

[38] H. J. T. Suh, M. Simchowitz, K. Zhang, and R. Tedrake, *Do Differentiable Simulators Give Better Policy Gradients?* Aug. 2022. arXiv: 2202.00817 [cs].

[39] V. Makoviychuk *et al.*, *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*, Aug. 2021. arXiv: 2108.10470 [cs].
[40] 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*, Jun. 2021. arXiv: 2106.13281 [cs].
[41] *MuJoCo: A physics engine for model-based control — IEEE Conference Publication — IEEE Xplore*, https://ieeexplore.ieee.org/document/6386109.
[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, Aug. 2017. arXiv: 1707.06347 [cs].
[43] C. Pan, Z. Yi, G. Shi, and G. Qu, *Model-Based Diffusion for Trajectory Optimization*, May 2024. arXiv: 2407.01573 [cs, eess, math].
[44] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.
[45] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi, "Theoretical Foundation for CMA-ES from Information Geometry Perspective," *Algorithmica*, vol. 64, no. 4, pp. 698–716, Dec. 2012.
[46] J. Bradbury *et al.*, *Google/jax*, Google, Sep. 2024.

## APPENDIX

### A. Hardware and Software Setup

We discuss various hardware platforms, including quadrupeds and humanoids, and provide details on the implementation of the algorithm.

**Quadraped Configurations** For quadrupedal tasks, we use a Unitree GO2 robot with 18 degrees of freedom (DoF) and 12 actuated joints. The abduction and hip joints can produce $24\,\mathrm{N}\cdot\mathrm{m}$ torque, and the knee joints can produce $45\,\mathrm{N}\cdot\mathrm{m}$ torque approximately. We perform basic system identification to match the joint response to that in the simulation as closely as possible. The low-level controller for each joint of the quadruped plays a crucial role across various tasks. To ensure fair comparisons, the PID controller parameters are kept consistent across different testbeds. In the GCRL reference implementation, which uses position control, we set the proportional gain to 30 and the derivative gain to 0.65. In our method, which utilizes torque control, we apply the same derivative gain of 0.65 to regulate excessively high motor speeds. This choice also helps stabilize the simulation, which uses the same derivative gain. Given a torque command $\tau_i$, the final torque command sent to the $i$-th joint is

$$\hat{\tau}_i = \tau_i - d\omega_i, \tag{8}$$

where $i$ is the index of the joint, $d = 0.65$ is the derivative gain and $\omega_i$ is the angular velocity of the $i$-th joint.

**Algorithm Implementation:** All algorithms are implemented in JAX [46], using the Brax simulator [40]. The GCRL model is trained using the reference PPO implementation provided by Brax. All inference runs are performed on a desktop equipped with an RTX 4090 GPU, i9-13900KF CPU, and 32 GB of RAM, with the computer communicating with the robot via an Ethernet connection. For real-time control, all algorithms operate at $50\,\mathrm{Hz}$, while a low-level controller repeats the last algorithm output and sends motor commands to the robot at $200\,\mathrm{Hz}$. Ground-truth state estimation is obtained via a motion capture system. All sampling-based MPCs utilize 2048 parallel environments and a 20-step horizon ($0.4\,\mathrm{s}$). To further speed up the simulations, contact forces are disabled for all robot parts except the feet.

| | Walking and Tracking | | Sequential Jumping | |
|---|---|---|---|---|
| | **DIAL-MPC** | **GCRL** | **DIAL-MPC** | **GCRL** |
| **Upright** | -1.0 | -1.0 | -1.0 | -1.0 |
| **Base Height** | -1.0 | -1.0 | -1.0 | -1.0 |
| **Energy** | -0.001 | -0.001 | -0.001 | -0.001 |
| **Linear Velocity** | -0.5 | -1.0 | - | - |
| **Angular Velocity** | -0.5 | -1.0 | - | - |
| **Gait** | -0.1 | -0.1 | - | - |
| **Contact Reward** | - | - | 0.1 | 0.1 |
| **Contact Penalty** | - | - | -0.1 | -0.1 |
| **Alive** | - | 3.0 | - | 10.0 |
| **Control Rate** | - | 0.001 | - | 0.001 |

TABLE IV: Reward specifications for the quadruped walking-tracking and sequential jumping tasks are largely shared between our method and the GCRL baseline, underscoring the flexibility of DIAL-MPC in utilizing RL-style rewards.

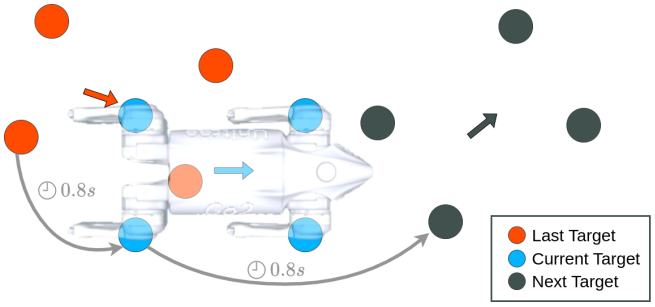


Fig. 8: A visual illustration of the sequential jumping task. At every stage, the quadruped is given $1\,\mathrm{s}$ to jump onto the next target. Contacts outside the target are penalized.

### B. Task Implementation Details

**Quadruped Velocity Tracking** We give all sampling-based MPCs the rewards specified in table IV. In particular, the gait is generated by a foot height generator outputting a trotting gait; energy is calculated as the product of joint torque $\tau_i$ and joint velocity $\omega_i$. With only six reward terms, DIAL-MPC demonstrates robust walking behavior in this task. We then applied GCRL using the same reward terms; to regularize the training process and improve the behavior, two additional key terms are introduced. The first is an alive reward to counterbalance the negative rewards and promote meaningful exploration during the initial stages of training. The second is a control rate cost, which subsumes the spline reparameterization trick used in DIAL-MPC to produce smoother actions. Furthermore, we implemented standard RL domain randomizations, varying the base mass by up to 1 kg, actuator gains by up to 20%, and the friction coefficient between 0.6 and 1.0. For the goal, we randomized the linear velocity target to $\pm(1.5, 0.5, 0.0)\ \mathrm{m/s}$, and the yaw angular velocity target to $\pm 1.5\ \mathrm{rad/s}$. We trained GCRL using PPO for 500 million steps, which took 31 minutes.

**Quadruped Sequential Jumping** In this task, quadruped

must rapidly jump onto a series of $10\,\mathrm{cm}$ plates. The task is depicted in Figure 8. At a regular time interval, a new contact target is given in terms of four-foot placements and a center-of-mass (CoM) location. The interval is so short ($1\,\mathrm{s}$) that the quadruped must transition smoothly and dynamically between the targets. We uniformly sample the next contact target, whose CoM location is within $\pm(0.65, 0.65, 0.0)\mathrm{m}$ translation and $0.5\,\mathrm{rad}$ yaw rotation w.r.t. the current contact target.

Inspired by WoCoCo [23], we use a staged contact reward system at time $t$:

$$r_{\text{con}}^{(j)}(t) = w_{\text{correct}} n_{\text{correct}}^{(j)}(t) - w_{\text{wrong}} \left( n_{\text{wrong}}^{(j)}(t) - n_{\text{correct}}^{(j-1)}(t) \right), \tag{9}$$

where $r_{\text{con}}^{(j)}$ is the current-stage contact reward function; $w_{\text{correct}}$ and $w_{\text{wrong}}$ are the contact reward and penalty weights; $n_{\text{correct}}^{(j)}$ and $n_{\text{wrong}}^{(j)}$ compute the correct and wrong number of contacts w.r.t. the current stage target; $n_{\text{correct}}^{(j-1)}$ computes the correct number of contacts w.r.t. the last stage target; The last term offsets the penalty of wrong contacts that were valid last-stage contacts. This prevents the robot from being penalized or rewarded while preparing for its current jump.

We establish a performance measurement using the total contact reward over a sequence of 10 jumping stages in $8\,\mathrm{s}$, denoted as $R_{\text{con}}$:

$$R_{\text{con}} = \sum_{j=1}^{j_{\max}} \min_{t \in [T_{\min}(j), T_{\max}(j))} \left( r_{\text{con}}^{(j)}(t) \right), \tag{10}$$

where $j_{\max} = 10$; $T_{\min}(j), T_{\max}(j)$ maps the time interval of the $j$-th contact stage; the minimum function finds the worst-case contact score at every stage.

We let every algorithm perform 5 trials, and report the average total contact reward, $\bar{R}_{\text{con}}$. Same as in the walking-tracking experiment, we add a $10\,\mathrm{kg}$ payload to the robot, and update the model parameters in all sampling-based MPCs accordingly.

**Quadruped Crate Climbing** In fig. 9, we demonstrate DIAL-MPC's ability to achieve RL-level agility in 1/15 the speed of real world. The quadruped is tasked with climbing onto the $0.6\,\mathrm{m}$ high crate, which is more than twice the standing height of the robot. Similar tasks have been demonstrated on several recent quadruped parkour works with RL [9], which report around 20 hours of training time for a single policy with perception. Although DIAL-MPC's integration with perception demands future work, it is currently comparable to a teacher policy with access to privileged environment information.

We use only five reward terms to guide DIAL-MPC to complete the task. They are shown in Table V. Contact reward is given to feet on the top surface of the crate. Since this is a non-real-time planning task, we use 4096 parallel samples, 40-step ($1\,\mathrm{s}$) horizon, and 4 annealing steps. We also enable all contacts on the base and thighs. We then roll out the simulation synchronously after DIAL-MPC finishes computing at every step, which in total takes 30 seconds to generate a 2-second full-state motion plan with zero prior knowledge or training.

| | Crate Climbing |
|---|---|
| **Target Position** | 0.5 |
| **Upright** | 0.01 |
| **Energy** | 0.001 |
| **Target Yaw** | 0.3 |
| **Contact Reward** | 0.2 |

TABLE V: Reward weights of the crate climbing task.

| | Humanoid Crate Pushing |
|---|---|
| **Gait** | 5.0 |
| **Upright** | 0.01 |
| **Yaw** | 0.1 |
| **Velocity** | 1.0 |
| **Torso Height** | 0.5 |
| **Energy** | 0.01 |
| **Contact Reward** | 0.05 |

TABLE VI: Reward weights of the humanoid crate pushing task.

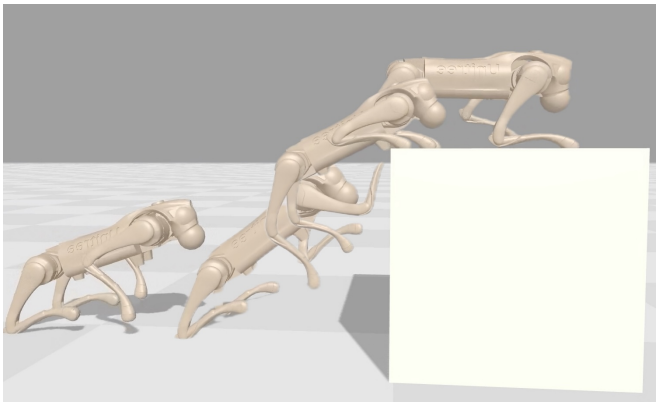

Fig. 9: DIAL-MPC generates this well-planned agile motion in 30 seconds, training free. The quadruped leaps forward to catch the edge of the crate with its forelegs, launches off the ground with its hindlegs, and quickly retracts to a standing pose on the crate.

**Humanoid Crate Pushing**

Figure 5 shows DIAL-MPC performing whole-body control task on a Unitree H1 humanoid in simulation. The robot is asked to track a slow velocity of $0.8\,\mathrm{m/s}$ while pushing forward a crate of $30\,\mathrm{kg}$ and $15\,\mathrm{kg}$ respectively. The robot is rewarded for maintaining the correct speed and the correct contact with its two upper end effectors. The seven reward weights are listed in Table VI. This task shows that DIAL-MPC generalizes well to systems with higher degrees of freedom (25 for the humanoid and 18 for the quadruped) and is capable of handling whole-body control tasks with versatile parameters. Again we highlight that all reward terms in this experiment are straight forward, and that it takes no time to visualize the outcome of reward tuning since DIAL-MPC is a training-free algorithm.