2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Adaptive Linear Path Model-Based Diffusion

Yutaka Shimizu and Masayoshi Tomizuka

Abstract— **The interest in combining model-based control approaches with diffusion models has been growing. Although we have seen many impressive robotic control results in difficult tasks, the performance of diffusion models is highly sensitive to the choice of scheduling parameters, making parameter tuning one of the most critical challenges. We introduce Linear Path Model-Based Diffusion (LP-MBD), which replaces the variance-preserving schedule with a flow-matching–inspired linear probability path. This yields a geometrically interpretable and decoupled parameterization that reduces tuning complexity and provides a stable foundation for adaptation. Building on this, we propose Adaptive LP-MBD (ALP-MBD), which leverages reinforcement learning to adjust diffusion steps and noise levels according to task complexity and environmental conditions. Across numerical studies, Brax benchmarks, and mobile-robot trajectory tracking, LP-MBD simplifies scheduling while maintaining strong performance, and ALP-MBD further improves robustness, adaptability, and real-time efficiency. Our code is available through anonymous repository `https://anonymous.4open.science/r/adaptive_linear_path_model_based_diffusion-C58C`**

I. INTRODUCTION

Diffusion models have achieved remarkable success in various real-world applications, particularly in image and video generation. Recently, their use has expanded beyond visual domains, with a growing body of work exploring applications in robotic control. Several approaches [1] [2] have been proposed that integrate diffusion models with model-based control to solve complex trajectory optimization problems. These approaches have close relationships with sampling-based optimization methods, and they have been shown to be effective in addressing problems that have nonlinear, non-smooth dynamics and non-convex objectives and constraints.

However, the performance of model-based diffusion critically depends on the design of the noise schedule. In diffusion-based approaches, the sampling process consists of a sequence of diffusion steps, where noise is incrementally injected into trajectories and progressively reduced to produce feasible and improved samples. The scale of the injected noise at each step plays an important role in determining performance, and even in the case of a simple linear parameterization with a variance-preserving schedule [1], we need to tune several parameters to get the optimal solution. These parameters are intricately coupled in shaping the overall noise profile, making it difficult to find optimal values. Moreover, the optimal parameters often vary with the system state and the complexity of the problem, which frequently leads to overly conservative parameter choices. Even within the same task, the values of the optimal parameters can differ substantially across different conditions. We illustrate this idea by using a self-driving car example shown in Fig. 1.

In this paper, we first introduce Linear Path Model-Based Diffusion (LP-MBD), a variant of the original Model-Based Diffusion (MBD) framework that replaces the variance-preserving schedule with a linear probability path inspired by Flow Matching [3][4]. LP-MBD offers several advantages over the original MBD. The linear probability path provides a simple, geometrically interpretable interpolation between prior and target distributions; under common Gaussian assumptions, it can align with optimal transport, offering clear theoretical grounding. More importantly, unlike the variance-preserving schedule, where multiple parameters are intricately coupled to determine the noise magnitude, the linear path formulation eliminates such dependencies. As a result, LP-MBD requires fewer hyperparameters, making the tuning process more straightforward and interpretable.

Building upon this foundation, we further propose Adaptive Linear Path Model-Based Diffusion (ALP-MBD), which extends LP-MBD with a reinforcement learning–based module for dynamic parameter adjustment. ALP-MBD adapts scheduling parameters according to task complexity and environmental conditions. For instance, it can increase the number of diffusion steps in challenging scenarios or enlarge the noise level to promote broader exploration, thereby enabling more flexible and effective control. In contrast, under simple or well-structured conditions, it can reduce both the noise magnitude and the diffusion steps, leading to more efficient sampling without sacrificing solution quality.

We extensively evaluate LP-MBD and ALP-MBD across diverse environments and settings to demonstrate their efficiency and performance. Our experiments include numerical studies, Mujoco-based benchmarks implemented in Brax, and a mobile robot trajectory-tracking task.

This paper has three main contributions:

1) **Linear Path Model-Based Diffusion (LP-MBD):** We introduce a flow-matching–inspired linear path scheduler that yields a geometrically interpretable and decoupled parameterization. This reduces the burden of tuning and provides a stable foundation for adaptive extensions.
2) **Adaptive Linear Path MBD (ALP-MBD):** Building on LP-MBD, we develop an adaptive scheduler that leverages reinforcement learning to adjust diffusion steps and noise levels based on the environment state, enhancing both robustness and efficiency.

Yutaka Shimizu and Masayoshi Tomizuka are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720, USA {purewater0901, tomizuka}@berkeley.edu

3) **Comprehensive evaluation:** We validate LP-MBD and ALP-MBD through numerical studies, Brax benchmarks, and mobile-robot trajectory tracking, demonstrating improved sample efficiency, control quality, and adaptability across diverse settings.

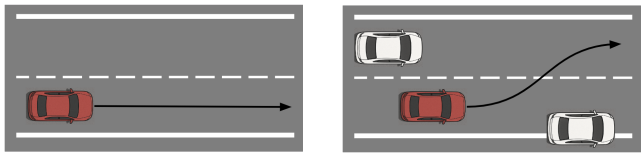

Fig. 1. The left figure illustrates a simple scenario in which the red ego vehicle drives in an obstacle-free environment. The right figure depicts a more complex case, where the red ego vehicle attempts to overtake a white vehicle while simultaneously avoiding an approaching car from behind. In the latter scenario, additional safety constraints are imposed, typically requiring more diffusion steps to obtain an optimal trajectory.

II. Related Work

Trajectory optimization is a fundamental component of safe and precise robot control. There are two principal approaches: (i) solving the underlying optimization problem directly with gradient-based methods [5], [6], [7], [8], and (ii) employing sampling-based approaches [9], [10] to search for optimal trajectories. This paper focuses on the latter, as it can better accommodate complex dynamics and objective functions. MPPI [11], [12] is among the most widely used sampling algorithms. It can be viewed as a soft-weighted variant of the Cross-Entropy Method (CEM) [13], where samples are weighted smoothly rather than selected by a hard elite set. CEM itself is closely related to CMA-ES [14], but CMA-ES further refines the update by adapting the covariance and step size in a principled way, making it more robust for problems with strong correlations or anisotropy.

Generative models provide powerful tools for capturing complex distributions. Diffusion models [15], [16], [17] and Flow Matching [3], [4] have gained significant attention due to their ability to represent intricate distributions and have been applied to planning problems [18], [19], [20], [21]. Recent works such as [22], [23] explore the link between sampling-based methods and diffusion models, while [2] highlights the connection between diffusion models and MPPI.

Reinforcement Learning (RL) [24] has been applied across diverse domains, including model-based planning in combination with sampling-based planners [25], [26], [27]. These studies demonstrate that integrating RL with sampling methods achieves superior performance compared to state-of-the-art model-free approaches [28], [29], [30], [31], [32]. Beyond planning, RL has also been used to fine-tune diffusion models [33], [34], [35]. For instance, [36] employs RL to optimize the initial noise in diffusion models, enabling mode selection within complex distributions. In this paper, we leverage RL to adaptively determine scheduling parameters conditioned on the environment state.

III. Linear Path Model-Based Diffusion

In this section, we first review Model-Based Diffusion (MBD) [1] and discuss why the choice of diffusion schedule can become a key practical bottleneck. We subsequently present Linear Path MBD (LP-MBD), which incorporates the linear probability path from Flow Matching. By constructing an optimal transport interpolation between the prior and target distributions, this formulation not only simplifies hyperparameter tuning but also provides a more stable and geometrically interpretable trajectory generation process.

A. Variance Preserving Model-Based Diffusion

Trajectory optimization (TO) is a fundamental tool for steering a robot toward a desired goal. We consider a finite-horizon problem with states $x_t \in \mathbb{R}^{n_x}$ and controls $u_t \in \mathbb{R}^{n_u}$:

$$\min_{x_{1:T},\, u_{1:T}} \quad J(x_{1:T}; u_{1:T}) \tag{1a}$$

$$\text{s.t.} \quad x_{t+1} = f_t(x_t, u_t), \tag{1b}$$

$$g_t(x_t, u_t) \le 0, \quad t = 0, \dots, T-1, \tag{1c}$$

where J is a user-specified cost, f_t denotes the system dynamics, and g_t is constraints. Let $Y = [x_{1:T}; u_{1:T}]$ denote the decision variables. Following [1], we recast TO as a sampling problem from the following target probability distribution

$$p_0(Y) \;\propto\; p_d(Y)\, p_J(Y)\, p_g(Y), \tag{2}$$

where $p_d(Y)$ enforces dynamical feasibility, $p_g(Y)$ encodes constraints, and $p_J(Y) \propto \exp\big(-J(Y)/\lambda\big)$ biases toward low cost with temperature $\lambda > 0$.

MBD [1] samples decision variables using a diffusion process. A standard forward process gradually perturbs an initial distribution p_0 toward an isotropic Gaussian p_N with noise governed by a schedule $\{\alpha_i\}_{i=1}^N$:

$$Y^{(i)} = c_{i,0}\, Y^{(0)} + c_{i,1}\, \varepsilon, \tag{3}$$

$$c_{i,0} = \sqrt{\bar{\alpha}_i}, \quad c_{i,1} = \sqrt{(1-\bar{\alpha}_i)}, \quad \bar{\alpha}_i = \prod_{k=1}^{i} \alpha_k, \tag{4}$$

where $\varepsilon \sim \mathcal{N}(0, I)$. Here, $Y^{(0)}$ denotes a sample drawn from the initial distribution p_0, which corresponds to the original decision variables of the optimization problem before any perturbation is applied. Note that the coefficients $c_{i,0}$ and $c_{i,1}$ define the noise schedule of the forward process and thus critically influence the performance of the diffusion model.

From Eq. (3) and Eq. (4), if the variance of the initial distribution is standardized such that $\mathrm{Var}[Y^{(0)}] = 1$, the variance of $Y^{(i)}$ remains equal to one for all i. This invariance arises because the coefficients $c_{i,0}$ and $c_{i,1}$ are chosen to satisfy $c_{i,0}^2 + c_{i,1}^2 = 1$, thereby preserving the overall variance throughout the forward diffusion process. For this reason, the schedule $\{\alpha_i\}$ is referred to as a variance-preserving (VP) noise schedule. In practice, MBD uses a simple linear VP schedule in which β_i is interpolated linearly over $i = 1, \dots, N$ with $\beta_0 = 1.0 \times 10^{-4}$ and $\beta_1 = 1.0 \times 10^{-2}$, and $\alpha_i = 1 - \beta_i$. We denote this specific instantiation of MBD

as VP-MBD, in order to distinguish it from the proposed approach.

Unlike model-free diffusion planners that learn the score from data, MBD exploits known objectives and dynamics to estimate the score $\nabla_{Y^{(i)}} \log p_i(Y^{(i)})$ and performs a Monte-Carlo score-ascent–type denoising step:

$$\begin{aligned} Y^{(i-1)} &= \frac{c_{i-1,0}}{c_{i,0}}\Big(Y^{(i)} + c_{i,1}^2\, \nabla_{Y^{(i)}} \log p_i\big(Y^{(i)}\big)\Big) \\ &= \frac{1}{\sqrt{\alpha_i}}\Big(Y^{(i)} + (1-\bar{\alpha}_i)\, \nabla_{Y^{(i)}} \log p_i\big(Y^{(i)}\big)\Big) \end{aligned} \tag{5}$$

The score can be written (via Bayes' rule and the forward kernel) as an expectation over "clean" trajectories $Y^{(0)}$ sampled from a Gaussian proposal and reweighted by the model-based target:

$$\begin{aligned} &\nabla_{Y^{(i)}} \log p_i\big(Y^{(i)}\big) \;\approx \\ &-\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \underbrace{\frac{\sum_{Y^{(0)}\in\mathcal{Y}^{(i)}_{\text{VP-MBD}}} Y^{(0)}\, w(Y^{(0)})}{\sum_{Y^{(0)}\in\mathcal{Y}^{(i)}_{\text{VP-MBD}}} w(Y^{(0)})}}_{\text{(importance-weighted average)}} \end{aligned} \tag{6}$$

where $w(Y) = p_J(Y)\, p_g(Y)$ and $\mathcal{Y}^{(i)}_{\text{VP-MBD}}$ follows a Gaussian distribution:

$$\mathcal{Y}^{(i)}_{\text{VP-MBD}} \sim \mathcal{N}\Big(\frac{Y^{(i)}}{\sqrt{\bar{\alpha}_i}}, \frac{1-\bar{\alpha}_i}{\bar{\alpha}_i}\Big) \tag{7}$$

For TO, candidate $Y^{(0)} = [x_{1:T}; u_{1:T}]$ are made dynamically feasible by rolling out $x_{t+1} = f_t(x_t, u_t)$ (shooting), and then scored via $w(Y)$. Substituting Eq. (6) into Eq. (5), we get

$$Y^{(i-1)} = \sqrt{\bar{\alpha}_{i-1}} \frac{\sum_{Y^{(0)}\in\mathcal{Y}^{(i)}_{\text{VP-MBD}}} Y^{(0)}\, w(Y^{(0)})}{\sum_{Y^{(0)}\in\mathcal{Y}^{(i)}_{\text{VP-MBD}}} w(Y^{(0)})} \tag{8}$$

Although VP-MBD attains strong performance, tuning the noise scheduling parameters remains challenging. In the variance-preserving (VP) formulation with simple linear scheduling, the schedule is specified by the triplet (β_0, β_1, T), which jointly determine $\{\alpha_i\}$ and the cumulative noise levels $\{\bar{\alpha}_i\}$. In particular, the maximum noise level is not controlled solely by the endpoints β_0 and β_1, but is also influenced by the total number of diffusion steps T. As a result, these parameters interact in a nontrivial manner to shape the effective noise scale at each step. This interdependence complicates the tuning process, as the impact of each parameter on the exploration–refinement trade-off is highly indirect and task-dependent, often requiring extensive trial-and-error to obtain satisfactory performance.

B. Linear Path Model-Based Diffusion

Motivated by flow matching, we adopt a *linear probability path* between a clean trajectory $Y^{(0)}$ and a standard Gaussian $\varepsilon \sim \mathcal{N}(0, I)$:

$$Y_t \;=\; (1-t)\, Y^{(0)} \;+\; t\, \varepsilon, \qquad t \in [0,1]. \tag{9}$$

Discretizing t on a uniform grid $t_i = \frac{i}{N}$, $i = 0, \ldots, N$, yields a schedule that increases the noise level linearly in "time."

$$Y^{(i)} \;=\; (1-t_i)\, Y^{(0)} \;+\; t_i\, \varepsilon \tag{10}$$

This corresponds to Eq. (3) when $c_{i,0} = 1 - t_i$ and $c_{i,1} = t_i$. At the endpoints, $i = 0$ ($t_i = 0$) gives $Y^{(i)} = Y^{(0)}$ (clean trajectory), while $i = N$ ($t_i = 1$) yields $Y^{(N)} = \varepsilon$ (standard Gaussian noise). Thus, the linear probability path recovers the clean sample at one endpoint and pure noise at the other, providing a clear and interpretable interpolation between the two distributions.

We use the same Monte-Carlo score-ascent-type denoising step:

$$Y^{(i-1)} = \frac{1-t_{i-1}}{1-t_i}\Big(Y^{(i)} + t_i^2\, \nabla_{Y^{(i)}} \log p_i\big(Y^{(i)}\big)\Big) \tag{11}$$

where the score function is

$$\begin{aligned} &\nabla_{Y^{(i)}} \log p_i\big(Y^{(i)}\big) \;\approx \\ &-\frac{Y^{(i)}}{t_i^2} + \frac{1-t_i}{t_i^2} \underbrace{\frac{\sum_{Y^{(0)}\in\mathcal{Y}^{(i)}_{\text{LP-MBD}}} Y^{(0)}\, w(Y^{(0)})}{\sum_{Y^{(0)}\in\mathcal{Y}^{(i)}_{\text{LP-MBD}}} w(Y^{(0)})}}_{\text{(importance-weighted average)}} \end{aligned} \tag{12}$$

Similar to VP-MBD, we sample $Y^{(0)}$ from the following Gaussian distribution

$$\mathcal{Y}^{(i)}_{\text{LP-MBD}} \sim \mathcal{N}\Big(\frac{Y^{(i)}}{1-t_i}, \frac{t_i^2}{(1-t_i)^2}\Big) \tag{13}$$

By substituting Eq. (12) into Eq. (11), we get

$$Y^{(i-1)} = (1-t_{i-1}) \frac{\sum_{Y^{(0)}\in\mathcal{Y}^{(i)}_{\text{LP-MBD}}} Y^{(0)}\, w(Y^{(0)})}{\sum_{Y^{(0)}\in\mathcal{Y}^{(i)}_{\text{LP-MBD}}} w(Y^{(0)})} \tag{14}$$

Although LP-MBD uses an intuitive noise schedule, the Gaussian proposal in Eq. (13) has a standard deviation $\sigma_i = \frac{t_i}{1-t_i}$, which diverges as $t_i \to 1$. This implies that the initial backward denoising step samples the entire trajectory space, which is theoretically valid but practically unnecessary. In trajectory optimization, control inputs are typically bounded by system constraints, so sampling from an unbounded domain is inefficient. Constraining the noise schedule to natural limits improves efficiency without loss of correctness, motivating a bounded scheduling strategy. Instead of extending the interpolation to $t_i = 1.0$, we truncate the schedule at a maximum value $t_{\max} < 1$, ensuring that the variance of $\mathcal{Y}^{(i)}_{\text{LP-MBD}}$ remains finite:

$$t_i \in [0,\, t_{\max}], \qquad t_{\max} < 1. \tag{15}$$

When $t_i = t_{\max}$, the standard deviation of the Gaussian distribution $\mathcal{Y}^{(i)}_{\text{LP-MBD}}$ reaches its maximum value.

$$\sigma_{\max} \;=\; \frac{t_{\max}}{1-t_{\max}} \quad \left(\text{equivalently, } t_{\max} = \frac{\sigma_{\max}}{1+\sigma_{\max}}\right) \tag{16}$$

In practice, $\sigma_{\max}$ can be determined from the admissible range of control inputs, which provides a direct and interpretable way to set the exploration limit. Therefore, $t_{\max}$ can be computed from Eq. (16).

C. Differences between VP-MBD and LP-MBD

The primary distinction between VP-MBD and LP-MBD lies in their parameterization and the implications for tuning and adaptation. In the VP-MBD formulation, the noise schedule is governed by three parameters (β_0, β_1, T), where the parameters are tightly coupled: modifying the diffusion horizon T not only changes the discretization but also increases or decreases the maximum noise variance. This interdependence complicates the tuning process and makes it difficult to isolate the effect of each parameter on exploration and refinement.

In contrast, LP-MBD is characterized by only two parameters: the maximum noise scale $\sigma_{\max}$ and the number of diffusion steps T. These parameters are *decoupled* and possess clear geometric interpretations: $\sigma_{\max}$ directly sets the maximum variance of the Gaussian proposal, while T controls the resolution of the interpolation. Importantly, adjusting T does not affect the extrema of the noise scale. This structural simplicity makes the noise schedule not only easier to tune but also more geometrically interpretable, providing an intuitive understanding of the interpolation between clean trajectories and noise. Beyond simplifying tuning, this decoupling also enables the adaptive extension in the next section. With two independent parameters, the parameter adaptation approach can efficiently optimize them, making LP-MBD a solid foundation for ALP-MBD.

IV. Adaptive Linear Path Model-Based Diffusion

In VP-MBD, the diffusion steps and noise scheduling parameters are fixed once selected and remain unchanged during execution. In practice, however, the difficulty of trajectory optimization can vary substantially even within the same task. For instance, navigation in open space may require a few diffusion steps, while obstacle-rich scenarios demand more refinement and broader exploration. This motivates adapting the diffusion process online to balance robustness and efficiency.

The decoupled and geometrically interpretable parameterization of LP-MBD provides a natural foundation for such adaptation. Unlike VP-MBD, where the parameters (β_0, β_1, T) are tightly coupled and changes in T also influence the effective noise scale, LP-MBD separates the maximum variance $\sigma_{\max}$ from the diffusion horizon T. This decoupling ensures that adjusting T only refines the discretization without unintentionally changing the exploration range. As a result, the training process of learning optimal parameters of LP-MBD is more stable and simple. In contrast, applying RL to VP-MBD would face instability, since parameter updates may produce unpredictable changes in the underlying noise profile. Thus, LP-MBD is inherently more compatible with adaptive parameter learning, enabling the design of Adaptive LP-MBD (ALP-MBD).

A. Formulation as a Reinforcement Learning Problem

Reinforcement learning (RL) aims to optimize decision-making policies in environments modeled as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0, \gamma)$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the transition dynamics, $\mathcal{R}$ the reward function, ρ_0 the initial state distribution, and γ the discount factor. At each time step, the agent observes $s \in \mathcal{S}$, selects an action $a \in \mathcal{A}$, transitions to $s' \sim \mathcal{P}(\cdot|s, a)$, and receives a reward $r(s, a)$.

In our formulation, the state space $\mathcal{S}$ corresponds to the environment state as in standard RL. However, unlike conventional RL where actions directly control the system, the action space $\mathcal{A}$ consists of the noise scheduling parameters T and $\sigma_{\max}$. The actual control signal u applied to the system is subsequently generated by LP-MBD with these estimated parameters. Accordingly, the adaptive noise scheduler serves as a policy that, at each step t, outputs

$$\begin{aligned} (T_t,\ \sigma_{\max,t}) &\sim \pi_\phi(\cdot \mid s_t), \\ \pi_\phi(T, \sigma_{\max} \mid s) &= \pi_T(T \mid s)\, \pi_\sigma(\sigma_{\max} \mid s), \end{aligned} \tag{17}$$

where ϕ denotes the policy parameters, π_T is a categorical distribution over $\mathcal{T} = \{T_{\min}, \ldots, T_{\max}\}$, and π_σ is a Gaussian distribution. The action at time step t is therefore defined as $a_t = (T_t, \sigma_{\max,t})$.

To promote efficiency, we augment the reward with a penalty on large diffusion steps:

$$\tilde{r}_t(s_t, a_t, u_t) = r(s_t, u_t) - w_T \frac{T_t}{T_{\max}}, \tag{18}$$

where $r(s_t, u_t)$ is the original environmental reward and $w_T > 0$ balances task performance and computational cost. Denoting LP-MBD as $\pi_{\text{LP-MBD}}(\cdot \mid s_t, a_t)$, the RL objective is

$$\max_\phi J(\phi) = \mathbb{E}_{\substack{s_t \sim \mathcal{B},\, a_t \sim \pi_\phi(\cdot|s_t), \\ u_t \sim \pi_{\text{LP-MBD}}(\cdot|s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t \tilde{r}_t \right], \tag{19}$$

with discount factor $\gamma \in (0, 1)$ and replay buffer $\mathcal{B}$. Fig. 2 shows the architecture of the proposed adaptive noise scheduling system.

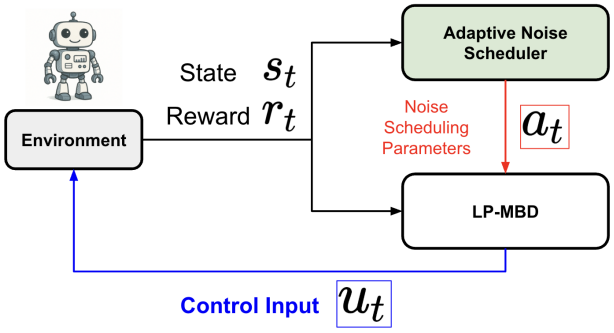

Fig. 2. Overview of ALP-MBD. The environment provides the current state s_t and reward r_t to the adaptive noise scheduler, which outputs the noise scheduling parameters $a_t = (T_t, \sigma_{\max,t})$. These parameters are fed into LP-MBD to generate the control input u_t, which is applied to the environment.

B. Implementation Details

To train the adaptive linear path scheduler, we employ the Proximal Policy Optimization (PPO) [37] algorithm. PPO is chosen for its stability in on-policy training and its

widespread adoption in both control and discrete tasks. Nevertheless, the proposed framework is agnostic to the choice of reinforcement learning algorithm, and other approaches (e.g., DDPG[31], SAC[29][38][39], or TD3[32]) can also be applied without loss of generality.

We summarize the training procedure in Algorithm 1. At each iteration, the adaptive noise scheduler samples scheduling parameters $(T_t, \sigma_{\max,t})$ according to the current state, which are then used by the LP-MBD to generate the control input u_t. The environment returns the next state s_{t+1} and reward r_t, which is reshaped into $\tilde{r}_t$ to penalize large diffusion steps. The PPO algorithm then updates the policy parameters ϕ and value function parameters ψ using collected trajectories from a rollout buffer.

Algorithm 1 Training ALP-MBD with PPO

Initialize policy parameters ϕ, value function parameters ψ
for each iteration **do**
  for each environment step t **do**
    Observe state s_t
    Sample scheduling parameters $a_t = (T_t, \sigma_{\max,t}) \sim \pi_\phi(\cdot|s_t)$
    Generate control input $u_t \sim \pi_{\text{LP-MBD}}(\cdot|s_t, a_t)$
    Apply u_t to environment, receive (s_{t+1}, r_t)
    Compute modified reward Eq. (18)
    Store $(s_t, a_t, u_t, \tilde{r}_t, s_{t+1})$ in buffer $\mathcal{B}$
  end for
  Use PPO update to optimize ϕ, ψ with data from $\mathcal{B}$
  Clear buffer $\mathcal{B}$
end for

V. Experiments

In this section, we conduct a series of experiments to evaluate the proposed LP-MBD and ALP-MBD. These experiments are designed to validate the following aspects: (i) the behavioral and performance differences between VP-MBD and LP-MBD; (ii) the advantages of LP-MBD in reducing the tuning burden; and (iii) the ability of ALP-MBD to adjust key parameters—primarily the diffusion horizon T and the maximum noise standard deviation $\sigma_{\max}$—as functions of the observed state, and the extent to which these adaptations improve efficiency. All experiments were conducted on a single NVIDIA RTX 4090 GPU. Throughout this experiment, we use $\beta_0 = 0.0001$ and $\beta_1 = 0.01$ for VP-MBD and $\sigma_{\max} = 1.8$ for LP-MBD.

A. Numerical Experiments

We first perform simple numerical experiments to evaluate LP-MBD and ALP-MBD and compare with VP-MBD, illustrating their relative advantages in a controlled setting. Our numerical studies (i) show why VP-MBD parameter tuning is harder than LP-MBD and (ii) illustrate how LP-MBD's simple and independent parameter sets better support adaptive scheduling.

Fig. 3 presents three one-dimensional examples. In the first example from [1], both VP-MBD and LP-MBD reach the optimal solution within 20 steps. However, in the second and third Gaussian objectives, VP-MBD fails to converge, while LP-MBD rapidly reaches the optimum in only a few steps. The performance gap between VP-MBD and LP-MBD can be attributed to differences in their noise scheduling. For instance, consider the second example with diffusion steps $T = 2$, where we compare the standard deviations of the Gaussian proposals for VP-MBD (Eq. (7)) and LP-MBD (Eq. (13)):

$$\begin{aligned} &\text{VP-MBD} \quad \sqrt{\frac{1-\bar{\alpha}_i}{\bar{\alpha}_i}} = [0.1,\, 0.01] \quad (i = 1, 0), \\ &\text{LP-MBD} \quad \frac{t_i}{1-t_i} = [1.8,\, 0.0] \quad (i = 1, 0). \end{aligned} \tag{20}$$

In this case, the maximum standard deviation of LP-MBD is $\sigma_{\max} = 1.8$, a value that remains unchanged regardless of the diffusion horizon T since $\sigma_{\max}$ is determined independently. By contrast, VP-MBD with the same parameters as in the first example ($\beta_0 = 10^{-2}$, $\beta_1 = 10^{-4}$) yields a Gaussian proposal distribution with extremely small variance, thereby suppressing stochasticity and precluding meaningful exploration. A similar issue arises in the third example with the Gaussian mixture objective and diffusion step $T = 5$, where LP-MBD again attains a maximum standard deviation of 1.8, while VP-MBD is limited to only 0.16. Consequently, the change of diffusion step T in VP-MBD can drastically influence its noise scale, necessitating additional tuning of β_0 and β_1 to match the behavior of LP-MBD. This comparison highlights that the proposed LP-MBD provides a more straightforward and intuitive parameterization, simplifying the tuning process relative to VP-MBD. Note that we only change the diffusion step T and keep using the same β_0 and β_1 for VP-MBD and $\sigma_{\max}$ for LP-MBD in these three examples.

We next evaluate ALP-MBD on a simple two-dimensional objective function. Using REINFORCE [40], we jointly optimize the continuous noise cap $\sigma_{\max}$ and the diffusion horizon T, with a penalty weight $w_T = 1.0$ on the step count to discourage unnecessarily large T. For comparison, we apply the same reinforcement learning procedure to VP-MBD—optimizing (β_0, β_1, T)—and refer to this baseline as Adaptive VP-MBD (AVP-MBD). For fair comparison, both methods are trained for the same number of policy gradient updates (30) under identical hyperparameters and evaluation budgets, and we evaluate each method using its learned parameters.

Figure 4 reports results on a 2D Gaussian mixture objective subject to a linear constraint. At $T = 3$, AVP-MBD (top) remains comparatively diffuse, with samples spread across both modes, whereas ALP-MBD (bottom) rapidly concentrates probability mass in the high-value feasible region, forming a compact cluster by step $T = 3$. These results indicate that ALP-MBD successfully discover an optimal parameter pair $(\sigma_{\max}, T)$ that improves both convergence speed and solution quality, while AVP-MBD

converges more slowly under the same training budget, due to the stronger parameter coupling among (β_0, β_1, T) and the larger hyperparameter search space.

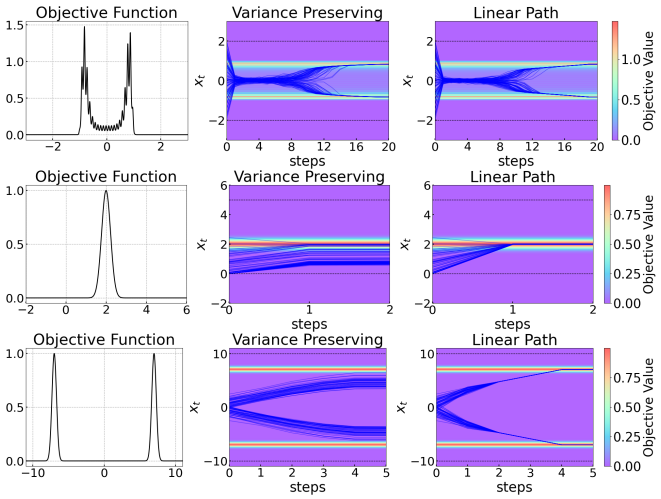

Fig. 3. 1D examples. (Top) The same example as in [1]. (Middle) A simple Gaussian objective function. (Bottom) A Gaussian mixture objective with two modes.

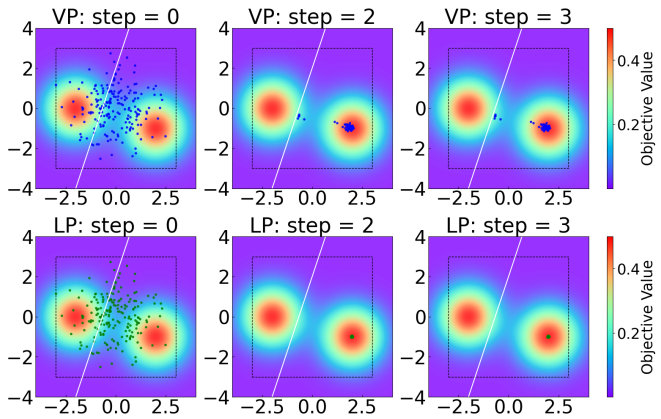

Fig. 4. The comparison of AVP-MBD and ALP-MBD. The white line represents the constraint $3x_0 - x_1 \geq 2.0$. The top row shows the result of VP-MBD, and the bottom row shows the results of ALP-MBD with estimated parameters. After 30 training steps, we get $\beta_0 = 0.000028$, $\beta_1 = 0.361$, $T = 3$ for AVP-MBD and $\sigma_{\max} = 2.11$, $T = 3$ for ALP-MBD.

B. Evaluation of LP-MBD

We evaluate the performance of LP-MBD on a range of tasks in the Brax environments [41]. Brax is a physics-based simulator designed for large-scale reinforcement learning research, providing fast and differentiable dynamics for a variety of continuous control tasks such as locomotion, manipulation, and navigation.

Table I summarizes the per-step rewards obtained by CEM [13], MPPI[11][12], VP-MBD[1], and the proposed LP-MBD. For a fair comparison, we use the same diffusion steps T, horizon H, and sampling number. Each value represents the mean performance and its standard deviation over 5 random seeds. Overall, LP-MBD achieves competitive or superior performance across most tasks. For instance, in Hopper, HalfCheetah, and Walker2D, our approach attains the highest rewards, highlighting its effectiveness in improving control quality.

However, it is worth noting that in the Pusher environment, LP-MBD has a lower score than VP-MBD. We hypothesize that this discrepancy arises from the higher dimensionality and increased complexity of the Pusher task. In such settings, variance-preserving noise scheduling may provide more expressive modeling capacity than a simple linear probability path, thereby yielding improved performance. We leave a more in-depth study to future work.

TABLE I

PER-STEP REWARDS FOR DIFFERENT TASKS.

Tasks	CEM	MPPI	VP-MBD	LP-MBD
Ant	**3.80 ± 0.43**	2.06 ± 0.44	3.67 ± 0.29	3.76 ± 0.15
Hopper	2.24 ± 0.05	2.31 ± 0.05	2.74 ± 0.03	**2.74 ± 0.01**
HalfCheetah	1.65 ± 2.33	2.10 ± 0.11	2.53 ± 0.31	**2.73 ± 0.12**
Walker2D	2.07 ± 0.02	2.08 ± 0.04	2.31 ± 0.02	**2.36 ± 0.02**
Reacher	-0.28 ± 0.04	-0.81 ± 0.06	**-0.17 ± 0.04**	**-0.17 ± 0.04**
Pusher	-0.95 ± 0.11	-3.47 ± 0.26	**-0.53 ± 0.11**	-0.74 ± 0.1

C. Evaluation of ALP-MBD

Finally, we evaluate ALP-MBD on a mobile robot trajectory-tracking task implemented in a custom gym-like environment. The robot is modeled using a kinematic vehicle model with state $x = [x, y, \theta, v]$, where (x, y) denotes the position, θ is the yaw angle, and v is the velocity. The control inputs are acceleration a and steering rate w, and the dynamics evolve as

$$\begin{aligned} x_{k+1} &= x_k + v_k \cos(\theta_k)\Delta t, \\ y_{k+1} &= y_k + v_k \sin(\theta_k)\Delta t, \\ \theta_{k+1} &= \theta_k + w_k \Delta t, \\ v_{k+1} &= v_k + a_k \Delta t, \end{aligned} \tag{21}$$

with time step Δt. The goal is to track a predefined reference path under these dynamics while ensuring safe and smooth behavior. To quantify performance, we use a reward function that penalizes deviations from the reference trajectory, misalignment in heading, velocity error, and collisions:

$$\begin{aligned} r = &-w_{\text{lat}}\, d_{\text{lat}} - w_{\text{yaw}}\, (\theta - \theta_{\text{ref}})^2 \\ &- w_v\, (v - v_{\text{ref}})^2 - w_{\text{collision}}\, \mathbf{1}_{\text{collision}}, \end{aligned} \tag{22}$$

where d_{lat} is the absolute lateral deviation from the reference path, ψ_{ref} and v_{ref} denote the desired yaw and velocity, $\mathbf{1}_{\text{collision}}$ is an indicator of collisions, and $w_\bullet$ denotes the weighting coefficients for each penalty term. The input state to the RL parameter tuning module is given by $(d_{\text{lat}}, d_\theta, d_{\text{vel}}, dx_{\text{obs}}, dy_{\text{obs}})$, where $d_\theta = \theta - \theta_{\text{ref}}$, $d_{\text{vel}} = v - v_{\text{ref}}$, and $(dx_{\text{obs}}, dy_{\text{obs}})$ denote the relative distances from the ego vehicle to surrounding obstacles. We train ALP-MBD in an environment with an S-shaped trajectory described in black line in Fig. 5. The gray rectangle in the figure shows a static obstacle in the environment. After the training, we compare the performance of ALP-MBD with VP-MBD and LP-MBD.

For fair comparison, we set the planning horizon $H = 50$ and the sampling number to 100 for all the algorithms. In this experiment, we implement each method using Python 3.11 and PyTorch.

Table II summarizes the average single-episode reward of each algorithm evaluated over five random seeds. Among the three MBD variants, the proposed adaptive method achieves the highest reward. For ALP-MBD, the reported diffusion steps T and maximum noise standard deviation $\sigma_{\max}$ are averaged across the five seeds. Regarding runtime, VP-MBD and LP-MBD measure only the time to generate a control input, whereas the ALP-MBD runtime also includes the parameter estimation process (a policy forward pass to determine T and $\sigma_{\max}$). This additional step accounts for its higher per-step latency relative to LP-MBD and VP-MBD. Nonetheless, the overhead introduced by parameter estimation is minor, and ALP-MBD remains well-suited for real-time control applications.

We also illustrate the trajectory generated by ALP-MBD in Fig. 5. ALP-MBD adaptively increases both diffusion steps and the maximum noise standard deviation when avoiding obstacles, while reducing them during steady cruising along the reference line. This adaptive behavior reflects the complexity of the underlying objective: when the ego vehicle is near an obstacle, the target objective function becomes more complex and requires additional iterations and a broader exploration range to converge to a feasible solution. In contrast, when the vehicle follows the reference trajectory in the absence of nearby obstacles, the objective remains simple, allowing convergence with fewer diffusion steps and a smaller variance. This adaptivity enhances the efficiency of VP-MBD by allocating greater computational effort only in challenging scenarios, while maintaining efficiency in simpler environments.

In addition, we evaluate the generalization ability of the trained ALP-MBD model in a new environment with a different reference trajectory. As shown in Fig. 6, ALP-MBD exhibits a consistent pattern with the previous results: it increases both the diffusion steps and the maximum noise standard deviation when the ego vehicle is near an obstacle, and decreases them when the vehicle is farther away. We further observe that the vehicle increases diffusion steps and the maximum standard deviation when driving sharp turns in the reference trajectory. In such cases, the vehicle must apply its maximum steering angle to remain aligned with the reference path, which necessitates both additional diffusion steps and a larger noise variance. Note that ALP-MBD does not see this trajectory during training, demonstrating its ability to generalize to previously unseen scenarios.

TABLE II

AVERAGE SINGLE EPISODE REWARD

Methods	Steps T	$\sigma_{\max}$	Reward	Runtime [ms]
VP-MBD	17	-	-3498 $\pm$ 116	38.3
LP-MBD	17	1.8	-3465 $\pm$ 147	**36.9**
ALP-MBD	16.9 $\pm$ 1.14	1.82 $\pm$ 0.18	**-3342 $\pm$ 128**	45.5

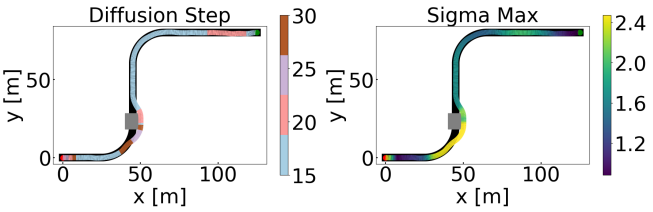

Fig. 5. ALP-MBD in trajectory following tasks for a mobile robot. The red point describes the start point, and the green point indicates the goal point. The black line shows the reference trajectory, and the gray rectangle is the obstacle.

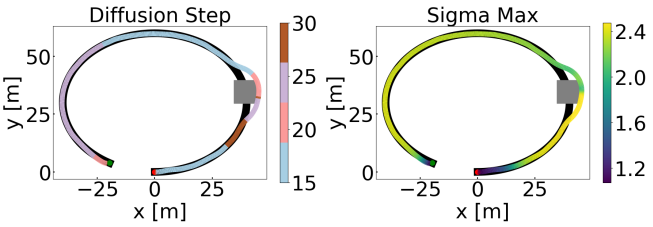

Fig. 6. ALP-MBD in trajectory following tasks for a mobile robot with an oval path. The red point describes the start point, and the green point indicates the goal point. The black line shows the reference trajectory, and the gray rectangle is the obstacle. We use the ALP-MBD trained in a different environment.

VI. CONCLUSIONS

We introduced Linear Path Model-Based Diffusion (LP-MBD) and its adaptive extension (ALP-MBD). LP-MBD replaces variance-preserving schedules with a flow-matching–inspired linear probability path, yielding a geometrically interpretable and decoupled parameterization. This not only reduces the burden of tuning but also provides the structural stability necessary for reinforcement learning–based adaptation. Building on this foundation, ALP-MBD dynamically adjusts diffusion steps and noise levels in response to task complexity, balancing robustness and efficiency. Through numerical examples, Brax benchmarks, and mobile robot trajectory-tracking, we demonstrated that LP-MBD simplifies scheduling while retaining strong performance, and that ALP-MBD further improves adaptability.

ACKNOWLEDGEMENT

This research was supported by TIER IV, Inc.

REFERENCES

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

[2] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4974–4981, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:272832507

[3] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=PqvMRDCJT9t

[4] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Q. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat, "Flow matching guide and code," 2024. [Online]. Available: https://arxiv.org/abs/2412.06264

[5] C. Hargraves and S. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987. [Online]. Available: https://doi.org/10.2514/3.20223

[6] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7674–7679.

[7] S.-J. Kim, K. Koh, M. Lustig, S. P. Boyd, and D. M. Gorinevsky, "An interior-point method for large-scale l1-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, pp. 606–617, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:675041

[8] G. Shi, W. Hönig, X. Shi, Y. Yue, and S.-J. Chung, "Neural-swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1063–1079, 2022.

[9] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[10] T. M. Howard, C. J. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *J. Field Robot.*, vol. 25, no. 6–7, p. 325–345, 2008.

[11] 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)*, 2016, pp. 1433–1440.

[12] ——, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[13] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Method. Comput. Appl. Prob.*, vol. 1, no. 2, p. 127–190, Sep. 1999. [Online]. Available: https://doi.org/10.1023/A:1010091220143

[14] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[15] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[16] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=St1giarCHLP

[17] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving latent diffusion models for high-resolution image synthesis," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=di52zR8xgf

[18] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[19] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2024.

[20] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*, 2022.

[21] X. Dai, D. Yu, S. Zhang, and Z. Yang, "Safe flow matching: Robot motion planning with control barrier functions," *CoRR*, vol. abs/2504.08661, April 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2504.08661

[22] J. Berner, L. Richter, and K. Ullrich, "An optimal control perspective on diffusion-based generative modeling," *Trans. Mach. Learn. Res.*, vol. 2024, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253255370

[23] V. Kurtz and J. W. Burdick, "Generative predictive control: Flow matching policies for dynamic and difficult-to-demonstrate tasks," 2025. [Online]. Available: https://arxiv.org/abs/2502.13406

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[25] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," in *ICML*, 2022.

[26] N. Hansen, H. Su, and X. Wang, "TD-MPC2: Scalable, robust world models for continuous control," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=Oxh5CstDJU

[27] Y. Shimizu and M. Tomizuka, "Bisimulation metric for model predictive control," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=F07ic7huE3

[28] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: contrastive unsupervised representations for reinforcement learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20. JMLR.org, 2020.

[29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1861–1870. [Online]. Available: https://proceedings.mlr.press/v80/haarnoja18b.html

[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, p. I–387–I–395.

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1509.02971

[32] J. Wu, Q. M. J. Wu, S. Chen, F. Pourpanah, and D. Huang, "A-td3: An adaptive asynchronous twin delayed deep deterministic for continuous action spaces," *IEEE Access*, vol. 10, pp. 128 077–128 089, 2022.

[33] M. Uehara, Y. Zhao, K. Black, E. Hajiramezanali, G. Scalia, N. L. Diamant, A. M. Tseng, S. Levine, and T. Biancalani, "Feedback efficient online fine-tuning of diffusion models," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.

[34] K. Clark, P. Vicol, K. Swersky, and D. J. Fleet, "Directly fine-tuning diffusion models on differentiable rewards," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=1vmSEVL19f

[35] Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. Lee, "Reinforcement learning for fine-tuning text-to-image diffusion models," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. [Online]. Available: https://openreview.net/forum?id=8OTPepXzeh

[36] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine, "Steering your diffusion policy with latent space reinforcement learning," *arXiv*, 2025.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms." *CoRR*, vol. abs/1707.06347, 2017.

[38] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *CoRR*, vol. abs/1812.05905, 2018. [Online]. Available: http://arxiv.org/abs/1812.05905

[39] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[40] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, p. 1057–1063.

[41] 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," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. [Online]. Available: https://openreview.net/forum?id=VdvDlnnjzIN