Model-Based Diffusion for Trajectory Optimization

Annoymous Authors

Abstract

Recent advances in diffusion models have demonstrated their strong capabilities in generating high-fidelity samples from complex distributions through an iterative refinement process. Despite the empirical success of diffusion models in motion planning and control, the model-free nature of these methods does not leverage readily available model information and limits their generalization to new scenarios beyond the training data (e.g., new robots with different dynamics). In this work, we introduce Model-Based Diffusion (MBD), an optimization approach using the diffusion process to solve trajectory optimization (TO) problems ***without data***. The key idea is to explicitly compute the score function by leveraging the model information in TO problems, which is why we refer to our approach as ***model-based*** diffusion. Moreover, although MBD does not require external data, it can be naturally integrated with data of diverse qualities to steer the diffusion process. We also reveal that MBD has interesting connections to sampling-based optimization. Empirical evaluations show that MBD outperforms state-of-the-art reinforcement learning and sampling-based TO methods in challenging contact-rich tasks. Additionally, MBD's ability to integrate with data enhances its versatility and practical applicability, even with imperfect and infeasible data (e.g., partial-state demonstrations for high-dimensional humanoids), beyond the scope of standard diffusion models. Videos and codes: `https://lecar-lab.github.io/mbd/`

1 Introduction

Trajectory optimization (TO) aims to optimize the state and control sequence to minimize a cost function while subject to specified dynamics and constraints. Given nonlinear, non-smooth dynamics and non-convex objectives and constraints, traditional optimization methods like gradient-based methods and interior point methods are less effective in solving TO problems. In response, diffusion models have emerged as a powerful tool for trajectory generation in complex dynamical systems due to their expressiveness and scalability [11, 46, 27, 26, 33, 5].

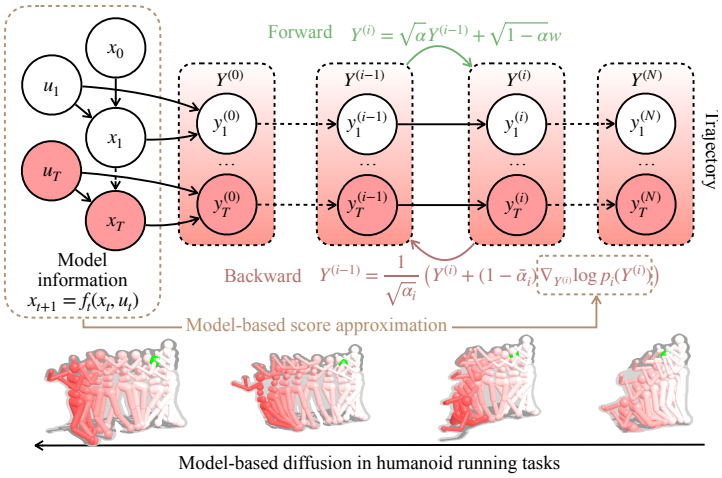

Figure 1: MBD refines the trajectory by leveraging the dynamics model directly without relying on demonstration data.

Although diffusion models excel when learning from large-scale, high-dimensional, and high-quality demonstrations, their dependency on such data limits their practicality. For example, after training a manipulation task with a specific robotic arm, the model may struggle to generalize to new tasks with a different arm as the underlying dynamics change. This limitation arises from the model-free nature

* Equal contributions. † Equal advising.

of existing diffusion-based methods, which do not leverage readily available model information to enhance adaptability. Moreover, existing diffusion-based approaches often require high-quality (in terms of optimality and feasibility) demonstration data, which limits their applications in various scenarios with imperfect data, such as dynamically infeasible trajectories (e.g., generated by high-level planners using simplified models) and partial demonstrations (e.g., lower-body-only demonstrations for a high-dimensional humanoid).

Fortunately, unlike diffusion model's applications in vision or language where data is from unknown distributions (e.g., internet-scale image data), in trajectory optimization, we often know the distribution of desired trajectories, which is described by the optimization objectives, constraints, and the underlying dynamics model, although such a distribution is intractable to directly sample from. Diffusion models offer a tantalizing new perspective, by iteratively refining samples from isotropic Gaussians to meaningful desired distributions in manageable steps, rather than directly learning the complex desired distribution. Inspired by this, we propose Model-Based Diffusion (MBD) that utilizes model information to approximate the gradient of the log probability density function (a.k.a. score function) and uses it to iteratively refine sampled trajectories to solve TO problems, as depicted in Fig. 1. This model-centric strategy allows for the generation of dynamically feasible trajectories in a data-free manner, and gradually moves them towards more optimal solutions. Furthermore, by using demonstrations as observations of the target distribution, MBD can be smoothly combined with data of different qualities to steer the diffusion process and enhance its effectiveness. Particularly, we merge the demonstration data into the sampling process by evaluating their likelihoods with the model and use them to improve the estimation of the score function. Our contributions are threefold:

- We introduce the Model-Based Diffusion (MBD) framework for trajectory optimization, utilizing the dynamics model to estimate the score function. This enables an effective trajectory planner given non-smooth dynamics and non-convex objectives, such as contact-rich manipulation tasks or high-dimensional humanoids.
- Our analysis and empirical evaluations demonstrate that MBD matches, and often exceeds, the performance of existing reinforcement learning and sampling-based TO methods. In particular, MBD outperforms PPO by 59% in various tasks within tens of seconds of diffusing.
- We demonstrate MBD's flexibility in utilizing diverse imperfect data to steer the diffusion process and further enhance the performance. Specifically, the resulting whole-body humanoid trajectory from MBD is more natural by utilizing the lower-body-state-only human motion data. Similarly, MBD can effectively address long-horizon sparse-reward Umaze navigation tasks by leveraging infeasible demonstrations generated by an RRT planner with simplified dynamics.

2 Related Work

Diffusion Models. Diffusion models have been widely adopted as generative models for high-dimensional data, such as image [43], audio [12], and text [7] through iterative refinement processes [42, 23]. The backward process can be viewed as gradient prediction [44] or score matching [45], which learns the score function to move samples towards the data distribution. We deliver new methods to perform the backward diffusion process using the available model information.

Sampling-based Optimization. Optimization involving black-box functions is widely applied across various fields, including hyperparameter tuning and experimental design [41, 22]. Evolutionary algorithms like CMA-ES are often used to tackle black-box optimization problems, dynamically modifying the covariance matrix to produce new samples [19]. Such problems can also be efficiently addressed within the Bayesian optimization framework [42, 15], which offers greater efficiency. Nonetheless, traditional BO algorithms are generally restricted to low-dimensional problems.

Trajectory Optimization. Traditionally, trajectory optimization (TO) is solved using gradient-based optimization, which faces challenges such as non-convex problem structures, nonlinear or discontinuous dynamics, and high-dimensional state and control action spaces. As two equivalent formulations, direct methods [20] and shooting-based methods [24] are commonly used to solve TO problems, where gradient-based optimizers such as Augmented Lagrangian [25], Interior Point [29], and Sequential Quadratic Programming [3, 40] are employed. To leverage the parallelism of modern hardware and improve global convergence properties, sampling-based methods like Cross-Entropy Motion Planning (CEM) [30] and Model Predictive Path Integral (MPPI) [49, 53] have been proposed to solve TO by sampling from target distributions. To solve stochastic optimal control problems, trajectory optimization has also been framed as an inference problem in a probabilistic graphical model, where system dynamics defines the graph structure [28, 32]. This perspective extends methods

such as iLQG by integrating approximate inference techniques to improve trajectory optimization [47]. The connection between diffusion and optimal control has been explored in [9], which motivates us to use diffusion models as solvers for trajectory optimization.

Diffusion for Planning. Diffusion-based planners have been used to perform human motion generation [11, 46] and multi-agent motion prediction [27]. Diffusion models are capable of generating complete trajectories by folding both dynamics and optimization processes into a single framework, thus mitigating compounding errors and allowing flexible conditioning [26, 33, 5]. In addition, they have been adeptly applied to policy generation, enhancing the capability to capture multimodal demonstration data in high-dimensional spaces for long-horizon tasks [38, 13]. These works assume no access to the underlying dynamics, limiting the generalization to new environments. To enforce dynamics constraints, SafeDiffuser [51] integrates control barrier functions into the learned diffusion process, while Diffusion-CCSP [52] composes the learned geometric and physical conditions to guarantee constraint compliance. Our approach uses diffusion models directly as solvers, rather than simply distilling solutions from demonstrations.

3 Problem Statement and Background

Notations: We use lower (upper) scripts to specify the time (diffusion) step, e.g., x_t, u_t, y_t represent the state, control and state-control pair at time t, and $Y^{(i)}$ represents the diffusion state at step i.

This paper focuses on a class of trajectory optimization (TO) problems whose objective is to find the sequences $\{x_t\}$ and $\{u_t\}$ that minimize the cost function $J(x_{1:T}; u_{1:T})$ subject to the dynamics and constraints. The optimization problem [1] can be formulated as follows:

$$\min_{x_{1:T}, u_{1:T}} J(x_{1:T}; u_{1:T}) = l_T(x_T) + \sum_{t=0}^{T-1} l_t(x_t, u_t) \tag{1a}$$

$$\text{s.t.} \quad x_0 = x_{\text{init}} \tag{1b}$$

$$x_{t+1} = f_t(x_t, u_t), \quad \forall t = 0, 1, \ldots, T-1, \tag{1c}$$

$$g_t(x_t, u_t) \le 0, \quad \forall t = 0, 1, \ldots, T-1. \tag{1d}$$

where $x_t \in \mathbb{R}^{n_x}$ and $u_t \in \mathbb{R}^{n_u}$ are the state and control at time t, $f_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ represents the dynamics, $g_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ are the constraint functions and $l_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ are the stage costs. We use $Y = [x_{1:T}; u_{1:T}]$ to denote all decision variables. Traditionally, TO is solved using nonlinear programming, which faces challenges such as non-convex problem structures, nonlinear or discontinuous dynamics, and high-dimensional state and control action spaces. Recently, there has been a growing interest in bypassing these challenges by directly generating samples from the optimal trajectory distribution using diffusion models trained on optimal demonstration data [11, 33, 38, 52].

To use diffusion for TO, (1) is first transformed into a sampling problem. The target distribution $p_0(Y^{(0)})$ is proportional to dynamical feasibility $p_d(Y) \propto \prod_{t=1}^{T} \mathbf{1}(x_t = f_{t-1}(x_{t-1}, u_{t-1}))$, optimality $p_J(Y) \propto \exp\left(-\frac{J(Y)}{\lambda}\right)$ and the constraints $p_g(Y) \propto \prod_{t=1}^{T} \mathbf{1}(g_t(x_t, u_t) \le 0)$, i.e.,

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

Obtaining the solution Y^* from the TO problem in Eq. (1) is equivalent to sampling from Eq. (2) given a low temperature $\lambda \to 0$. In fact, in Appendix A.1, we prove that the distribution of $J(Y)$ with $Y \sim p_0(\cdot)$ converges in probability to the optimal value J^* as $\lambda \to 0$, under mild assumptions. However, it is generally difficult to directly sample from the high-dimensional and sparse target distribution $p_0(\cdot)$. To address this issue, the diffusion process iteratively refines the samples following a backward process, which reverses a predefined forward process as shown in Fig. 1. The forward process corrupts the original distribution $p_0(\cdot)$ to an isotropic Gaussian $p_N(\cdot)$ by incrementally adding small noise to it and scaling it down by $\sqrt{\alpha_i}$ to maintain an invariant noise covariance (see Fig. 2(b) for an example). Mathematically, this means we iteratively obtain $Y^{(i)} \sim p_i(\cdot)$ with $p_{i|i-1}(\cdot|Y^{(i-1)}) \sim \mathcal{N}(\sqrt{\alpha_i} Y^{(i-1)}, \sqrt{1-\alpha_i} I)$. Because the noise at each time step is independent, the conditional distribution of $Y^{(i)}|Y^{(i-1)}$ also leads to that of $Y^{(i)}|Y^{(0)}$:

$$p_{i|0}(\cdot|Y^{(0)}) \sim \mathcal{N}(\sqrt{\bar{\alpha}_i} Y^{(0)}, \sqrt{1-\bar{\alpha}_i} I), \quad \bar{\alpha}_i = \prod_{k=1}^{i} \alpha_k. \tag{3}$$

[1] We assume deterministic dynamics for simplicity to sample the dynamically feasible trajectory. The extension to stochastic dynamics is straightforward.

The backward process $p_{i-1|i}(Y^{(i-1)}|Y^{(i)})$ is the reverse of the forward process $p_{i|i-1}(Y^{(i)}|Y^{(i-1)})$, which removes the noise from the corrupted distribution $p_N(\cdot)$ to obtain the target distribution $p_0(\cdot)$. The target distribution $p_0(\cdot)$ in the diffusion process is given by:

$$p_{i-1}(Y^{(i-1)}) = \int p_{i-1|i}(Y^{(i-1)}|Y^{(i)}) p_i(Y^{(i)})\, dY^{(i)}, \tag{4}$$

$$p_0(Y^{(0)}) = \int p_N(Y^{(N)}) \prod_{i=N}^{1} p_{i-1|i}(Y^{(i-1)}|Y^{(i)}) dY^{(1:N)} \tag{5}$$

Standard diffusion models [26, 33, 52], which we refer to as Model-Free Diffusion (MFD), solve the backward process by learning score function merely from data. In contrast, we propose leveraging the dynamics model to estimate the score to improve the generalizability of the model and allow a natural integration with diverse quality data.

4 Model-Based Diffusion

In this section, we formally introduce our MBD algorithm that leverages model information to perform backward process. To streamline the discussion, in Section 4.1, we first present MBD with Monte Carlo score ascent to solve simplified and generic unconstrained optimization problems. In Section 4.2, we extend MBD to the constrained optimization setting to solve the TO problem given complex dynamics and constraints. Lastly, in Section 4.3, we augment the MBD algorithm with demonstrations to improve sample quality and steer the diffusion process.

4.1 Model-based Diffusion as Multi-stage Optimization

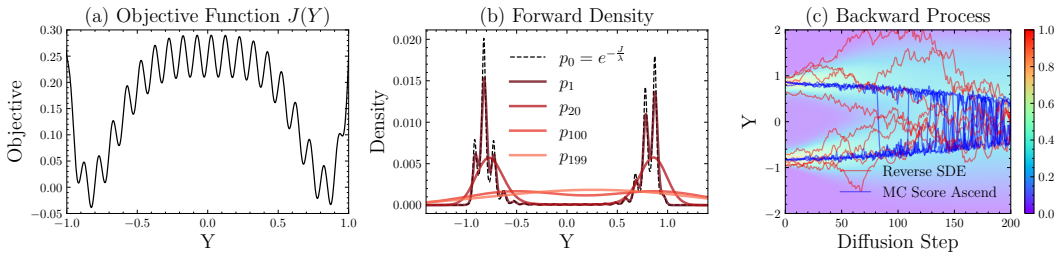

Figure 2: Reverse SDE vs. Monte Carlo score ascent (MCSA) on a synthetic highly non-convex objective function. (a) Synthesized objective function with multiple local minima. (b) The intermediate stage density $p_i(\cdot)$, where peaked $p_0(\cdot)$ is iteratively corrupted to a Gaussian $p_N(\cdot)$. (c) Reverse SDE vs. MCSA: Background colors represent the density of $p_i(\cdot)$ at different stages. MCSA converges faster due to larger step size and lower sampling noise while still capturing the multimodality.

We consider the reverse process for an unconstrained optimization problem $\min_Y J(Y)$, where the target distribution is $p_0(Y^{(0)}) \propto \exp(-\frac{J(Y^{(0)})}{\lambda})$. In our MBD framework, "model" implies that we can evaluate $J(Y^{(0)})$ for arbitrary $Y^{(0)}$, enabling us to compute the target distribution up to the normalizing constant.

MBD uses Monte Carlo score ascent instead of the commonly adopted reverse SDE approach in MFD. Specifically, when denoising from i to $i-1$, MBD performs one step of gradient ascent on $\log p_i(Y^{(i)})$ and then scales the sample by the factor $\frac{1}{\sqrt{\alpha_i}}$ as defined in the forward process:

$$Y^{(i-1)} = \frac{1}{\sqrt{\alpha_i}} \left(Y^{(i)} + (1-\bar{\alpha}_i) \nabla_{Y^{(i)}} \log p_i(Y^{(i)}) \right) \tag{6}$$

Critically, with the model-informed $p_0(Y^{(0)})$, we can estimate the score function $\nabla_{Y^{(i)}} \log p_i(Y^{(i)})$ by connecting $p_i(Y^{(i)})$ to $p_0(Y^{(0)})$ via Bayes' rule:

$$\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) = \frac{\nabla_{Y^{(i)}} \int p_{i|0}(Y^{(i)} \mid Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}}{\int p_{i|0}(Y^{(i)} \mid Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}} = \frac{\int \nabla_{Y^{(i)}} p_{i|0}(Y^{(i)} \mid Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}}{\int p_{i|0}(Y^{(i)} \mid Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}} \tag{7a}$$

$$= \frac{\int -\frac{Y^{(i)} - \sqrt{\bar{\alpha}_i} Y^{(0)}}{1-\bar{\alpha}_i} p_{i|0}(Y^{(i)} \mid Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}}{\int p_{i|0}(Y^{(i)} \mid Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}} \tag{7b}$$

$$= -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \frac{\int Y^{(0)} p_{i|0}(Y^{(i)} \mid Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}}{\int p_{i|0}(Y^{(i)} \mid Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}} \tag{7c}$$

Between Eq. (7a) and Eq. (7b), we use the forward Gaussian density in Eq. (3): $p_{i|0}(Y^{(i)} \mid Y^{(0)}) \propto \exp\left(-\frac{1}{2} \frac{\left(Y^{(i)} - \sqrt{\bar{\alpha}_i} Y^{(0)}\right)^\top \left(Y^{(i)} - \sqrt{\bar{\alpha}_i} Y^{(0)}\right)}{1-\bar{\alpha}_i}\right)$. Its log-likelihood gradient is $\nabla_{Y^{(i)}} p_{i|0}(Y^{(i)} \mid Y^{(0)}) = -\frac{1}{1-\bar{\alpha}_i}(Y^{(i)} - \sqrt{\bar{\alpha}_i} Y^{(0)}) p_{i|0}(Y^{(i)} \mid Y^{(0)})$. Given $Y^{(0)}$ as the integration variable in Eq. (7c), $p_{i|0}(Y^{(i)} \mid Y^{(0)})$ is evaluated as a function of $Y^{(0)}$ parameterized by $Y^{(i)}$. Based on that, we define the function $\phi_i(Y^{(0)})$ as:

$$\phi_i(Y^{(0)}) \propto p_{i|0}(Y^{(i)} \mid Y^{(0)}) \propto \exp\left(-\frac{1}{2} \frac{\left(Y^{(0)} - \frac{Y^{(i)}}{\sqrt{\bar{\alpha}_i}}\right)^\top \left(Y^{(0)} - \frac{Y^{(i)}}{\sqrt{\bar{\alpha}_i}}\right)}{\frac{1-\bar{\alpha}_i}{\bar{\alpha}_i}}\right) \propto \mathcal{N}\left(\frac{Y^{(i)}}{\sqrt{\bar{\alpha}_i}}, \frac{I}{\bar{\alpha}_i} - I\right) \tag{8}$$

This finding enables the Monte-Carlo estimation for computing the score function. We collect a batch of samples from $\phi_i(\cdot)$ which we denote as $\mathcal{Y}^{(i)}$ and approximate the score as:

$$\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) = -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \frac{\int Y^{(0)} \phi_i(Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}}{\int \phi_i(Y^{(0)}) p_0(Y^{(0)})\, dY^{(0)}} \tag{9a}$$

$$\approx -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \underbrace{\frac{\sum_{Y^{(0)} \in \mathcal{Y}^{(i)}} Y^{(0)} p_0(Y^{(0)})}{\sum_{Y^{(0)} \in \mathcal{Y}^{(i)}} p_0(Y^{(0)})}}_{\text{Monte Carlo Approximation}} := -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \bar{Y}^{(0)}(\mathcal{Y}^{(i)}) \tag{9b}$$

Aspect	Model-Based Diffusion (MBD)	Model-Free Diffusion (MFD)
Target Distribution	Known (Eq. (2)), but hard to sample	Unknown, but have data
Objective	Sample $Y^{(0)}$ from high-likelihood region of $p_0(\cdot)$	Sample $Y^{(0)} \sim p_0(\cdot)$
Score Approximation	Estimated using the model (Eq. (9a)). Can be augmented with demonstrations (Eqs. (11) and (13))	Learned from data
Backward Process	Perform Monte Carlo score ascent (Eq. (6)) to move samples towards most-likely states	Run reverse SDE to preserve sample diversity

Table 1: Comparison of Model-Based Diffusion (MBD) and Model-Free Diffusion (MFD)

Comparison between MFD and MBD. Table 1 highlights the key differences between MBD and MFD, which originate from two assumptions made in MBD: (a) a known target distribution $p_0(Y^{(0)})$ given the model; (b) the objective of sampling $Y^{(0)}$ from the high-likelihood region of $p_0(Y^{(0)})$ to minimize the cost function. For (a), MBD leverages p_0 to estimate the score following Eq. (9a), whereas MFD learns that from the data. For (b), MBD runs Monte Carlo score ascent in Eq. (6) to quickly move the samples to the high-density region as depicted in Fig. 2(c), while MFD runs reverse SDE $Y^{(i-1)} = \frac{1}{\sqrt{\alpha_i}}\left(Y^{(i)} + \frac{1-\alpha_i}{2} \nabla_{Y^{(i)}} \log p_i(Y^{(i)})\right) + \sqrt{1-\alpha_i}\mathbf{z}_i$, where $\mathbf{z}_i$ is Gaussian noise, to maintain the sample diversity. Given low temperature λ, it can be shown that $\nabla \log p_i(Y^{(i)}) \approx -\frac{1}{(1-\bar{\alpha}_i)}(Y^{(i)} - \arg\max p_i(\cdot))$ [2], i.e., the function $\log p_i(Y^{(i)})$ is $\frac{1}{(1-\bar{\alpha}_i)}$-smooth. Therefore, choosing the step size $(1-\bar{\alpha}_i)$ in Eq. (6) is considered optimal, as for L-smooth functions, $O(\frac{1}{L})$ is the step size that achieves the fastest convergence [55].

How diffusion helps? The diffusion process plays an important role in helping Monte Carlo score ascent overcome the local minimum issue in highly non-convex optimization problems, as shown in Fig. 2(a). Compared with optimizing a highly non-convex objective, Monte Carlo score ascent

[2] See more elaborations in Appendix A.1

is applied to the intermediate distribution $p_i(\cdot) = \int p_0(Y^{(0)})p_{i|0}(\cdot)dY^{(0)}$, which is made concave by convoluting $p_0(\cdot)$ with a Gaussian distribution $p_{i|0}(\cdot)$, as shown in Fig. 2(b). Starting from the strongly concave Gaussian distribution $p_N \sim \mathcal{N}(\mathbf{0}, I)$ with scale $\bar{\alpha}_N \to 0$, the density is easy to sample. The covariance of the sampling density $\Sigma_{\phi_i} = (\frac{1}{\bar{\alpha}_i} - 1)I$ is large when $i = N$, implying that we are searching a wide space for global minima. In the less-noised stage, the intermediate distribution $p_i(\cdot)$ is more peaked and closer to the target distribution $p_0(\cdot)$, and $\bar{\alpha}_i \to 1$ produces a smaller sampling covariance Σ_{ϕ_i} to perform a local search. By iteratively running gradient ascent on the smoothed distribution, MBD can effectively optimize a highly non-convex objective function as presented in Fig. 2. The MBD algorithm is formally depicted in Algorithm 1.

Algorithm 1 Model-based Diffusion for Generic Optimization

1: **Input:** $Y^{(N)} \sim \mathcal{N}(\mathbf{0}, I)$
2: **for** $i = N$ to 1 **do**
3: Sample $\mathcal{Y}^{(i)} \sim \mathcal{N}(\frac{Y^{(i)}}{\sqrt{\bar{\alpha}_{i-1}}}, (\frac{1}{\bar{\alpha}_{i-1}} - 1)I)$
4: Calculate Eq. (9b) $\bar{Y}^{(0)}(\mathcal{Y}^{(i)}) = \frac{\sum_{Y^{(0)} \in \mathcal{Y}^{(i)}} Y^{(0)} p_0(Y^{(0)})}{\sum_{Y^{(0)} \in \mathcal{Y}^{(i)}} p_0(Y^{(0)})}$
5: Estimate the score Eq. (9a): $\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) \approx -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \bar{Y}^{(0)}(\mathcal{Y}^{(i)})$
6: Monte Carlo score ascent Eq. (6): $Y^{(i-1)} \leftarrow \frac{1}{\sqrt{\alpha_i}} \left(Y^{(i)} + (1 - \bar{\alpha}_i) \nabla_{Y^{(i)}} \log p_i(Y^{(i)}) \right)$
7: **end for**

Connection with Sampling-based Optimization. When diffusion step is set to $N = 1$, MBD effectively reduces to the Cross-Entropy Method (CEM) [10] for optimization. To see this, we can plug the estimated score Eq. (9b) into the Monte Carlo score ascent Eq. (6) and set $N = 1$: $Y^{(0)} = \frac{\bar{\alpha}_1}{\alpha_1} \bar{Y}^{(0)}(\mathcal{Y}^{(1)}) = \bar{Y}^{(0)}(\mathcal{Y}^{(1)}) = \frac{\sum_{Y^{(0)} \in \mathcal{Y}^{(1)}} Y^{(0)} w(Y^{(0)})}{\sum_{Y^{(0)} \in \mathcal{Y}^{(1)}} w(Y^{(0)})}$ where $w(Y^{(0)}) = p_0(Y^{(0)}) \propto \exp(-\frac{J(Y^{(0)})}{\lambda})$ and $\mathcal{Y}^{(1)} \sim \mathcal{N}(\frac{Y^{(1)}}{\alpha_0}, (\frac{1}{\alpha_0} - 1)I)$. This precisely mirrors the update mechanism in CEM, which aims to optimize the objective function $f_{\text{CEM}}(Y^{(0)}) = J(Y^{(0)})$ and determine the sampling covariance $\Sigma_{\text{CEM}} = (\frac{1}{\alpha_0} - 1)I$, thus linking the sampling strategy of CEM with the α schedule in MBD. The advances that distinguish MBD from CEM-like methods are (1) the careful scheduling of α and (2) the intermediate refinements on p_i, both following the forward diffusion process. This allows MBD to optimize for smoothed functions in the early stage and gradually refine the solution for the original objective. On the contrary, CEM's solution could either be biased given a large Σ_{CEM} which overly smoothes the distribution as in p_{20}, p_{100} of Fig. 2(b), or stuck in local minima with a small Σ_{CEM} as in p_1 of Fig. 2(b) where the distribution is highly non-concave.

4.2 Model-based Diffusion for Trajectory Optimization

For TO, we have to accommodate the constraints in Eq. (1) which change the target distribution to $p_0(Y^{(0)}) \propto p_d(Y^{(0)})p_J(Y^{(0)})p_g(Y^{(0)})$. Given that $p_d(Y^{(0)})$ is a Dirac delta function that assigns non-zero probability only to dynamically feasible trajectories, sampling from $\phi_i(Y^{(0)})$ could result in low densities. To enhance sampling efficiency, we collect a batch of dynamically feasible samples $\mathcal{Y}_d^{(i)}$ from the distribution $\phi_i(Y^{(0)})p_d(Y^{(0)})$ with model information. Proceeding from Eq. (9a), and incorporating $p_0(Y^{(0)}) \propto p_d(Y^{(0)})p_J(Y^{(0)})p_g(Y^{(0)})$, we show the score function is:

$$\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) = -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \frac{\int Y^{(0)} \phi_i(Y^{(0)}) p_d(Y^{(0)}) p_g(Y^{(0)}) p_J(Y^{(0)}) \, dY^{(0)}}{\int \phi_i(Y^{(0)}) p_d(Y^{(0)}) p_g(Y^{(0)}) p_J(Y^{(0)}) \, dY^{(0)}} \tag{10a}$$

$$\approx -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \frac{\sum_{Y^{(0)} \in \mathcal{Y}_d^{(i)}} Y^{(0)} p_J(Y^{(0)}) p_g(Y^{(0)})}{\sum_{Y^{(0)} \in \mathcal{Y}_d^{(i)}} p_J(Y^{(0)}) p_g(Y^{(0)})} \tag{10b}$$

$$= -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \bar{Y}^{(0)}, \tag{10c}$$

$$\text{where} \quad \bar{Y}^{(0)} = \frac{\sum_{Y^{(0)} \in \mathcal{Y}_d^{(i)}} Y^{(0)} w(Y^{(0)})}{\sum_{Y^{(0)} \in \mathcal{Y}_d^{(i)}} w(Y^{(0)})}, \quad w(Y^{(0)}) = p_J(Y^{(0)}) p_g(Y^{(0)}) \tag{10d}$$

The model plays a crucial role in score esitimation by transforming infeasible samples $\mathcal{Y}^{(i)}$ from Line 3 in Algorithm 2 into feasible ones $\mathcal{Y}_d^{(i)}$. The conversion is achieved by putting the

control part $U = u_{1:T}$ of $Y^{(0)} = [x_{1:T}; u_{1:T}]$ into the dynamics Eq. (1c) recursively to get the dynamically feasible samples $Y_d^{(0)}$ (Line 4), which shares the same idea with the shooting method [24] in TO. MBD then evaluates the weight of each sample with $p_g(Y^{(0)})p_J(Y^{(0)})$ in Line 5. One common limitation of shooting methods is that they could be inefficient for long-horizon tasks due to the combinatorial explosion of the constrained space $p_g(Y) \propto \prod_{t=1}^{T} \mathbf{1}(g_t(x_t, u_t) \leq 0)$, leading to low constraint satisfaction rates. To address this issue, we will introduce demonstration-augmented MBD in Section 4.3 to guide the sampling process from the state space to improve sample quality.

Algorithm 2 Model-based Diffusion for Trajectory Optimization

1: **Input:** $Y^{(N)} \sim \mathcal{N}(\mathbf{0}, I)$
2: **for** $i = N$ to 1 **do**
3: Sample $\mathcal{Y}^{(i)} \sim \mathcal{N}(\frac{Y^{(i)}}{\sqrt{\bar{\alpha}_{i-1}}}, (\frac{1}{\bar{\alpha}_{i-1}} - 1)I)$
4: Get dynamically feasible samples: $\mathcal{Y}_d^{(i)} \leftarrow \text{rollout}(\mathcal{Y}^{(i)})$
5: Calculate $\bar{Y}^{(0)}$ with Eq. (10d) (model only) or Eq. (13) (model + demonstration)
6: Estimate the score Eq. (10c): $\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) \approx -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i}\bar{Y}^{(0)}$
7: Monte Carlo score ascent Eq. (6): $Y^{(i-1)} \leftarrow \frac{1}{\sqrt{\alpha_i}}\left(Y^{(i)} + (1-\bar{\alpha}_i)\nabla_{Y^{(i)}} \log p_i(Y^{(i)})\right)$
8: **end for**

4.3 Model-based Diffusion with Demonstration

With the ability to leverage model information, MBD can also be seamlessly integrated with various types of data, including imperfect or partial-state demonstrations by modeling them as noisy observations of the desired trajectory $p(Y_{\text{demo}} \mid Y^{(0)}) \sim \mathcal{N}(Y^{(0)}, \sigma^2 I)$. Given suboptimal demonstrations, sampling from the posterior $p(Y^{(0)} \mid Y_{\text{demo}}) \propto p_0(Y^{(0)})p(Y_{\text{demo}} \mid Y^{(0)})$ could lead to poor solutions as the demonstration likelihood $p(Y_{\text{demo}} \mid Y^{(0)})$ could dominate the model-based distribution $p_0(Y^{(0)}) \propto p_d(Y^{(0)})p_J(Y^{(0)})p_g(Y^{(0)})$ and mislead the sampling process. Rather, we assess $Y^{(0)}$ using $p(Y_{\text{demo}} \mid Y^{(0)})$, employing a similar technique to interchange the distribution's parameter with the random variable, as demonstrated in Eq. (8), to establish $p_{\text{demo}}(Y^{(0)}) \propto p(Y_{\text{demo}} \mid Y^{(0)}) \propto \mathcal{N}(Y^{(0)} \mid Y_{\text{demo}}, \sigma^2 I)$.

To accommodate demonstrations of varying qualities, instead of fixing target to $p_0(Y^{(0)})p(Y_{\text{demo}} \mid Y^{(0)})$, we propose seperating the $p_0(Y^{(0)})$ from $p_{\text{demo}}(Y^{(0)})$ to form a new target distribution[3]:

$$p_0'(Y^{(0)}) \propto (1-\eta)p_d(Y^{(0)})p_J(Y^{(0)})p_g(Y^{(0)}) + \eta p_{\text{demo}}(Y^{(0)})p_J(Y_{\text{demo}})p_g(Y_{\text{demo}}) \tag{11}$$

where η is a constant to balance the model and the demonstration. Here, we have introduced two extra constant terms $p_J(Y_{\text{demo}})$ and $p_g(Y_{\text{demo}})$ to ensure that the demonstration likelihood is properly scaled to match the model likelihood $p_0(Y^{(0)})$. With these preparations, we propose to adaptively determine the significance of the demonstration by choosing η as follows:

$$\eta = \begin{cases} 1 & p_d(Y^{(0)})p_J(Y^{(0)})p_g(Y^{(0)}) < p_{\text{demo}}(Y^{(0)})p_J(Y_{\text{demo}})p_g(Y_{\text{demo}}) \\ 0 & p_d(Y^{(0)})p_J(Y^{(0)})p_g(Y^{(0)}) \geq p_{\text{demo}}(Y^{(0)})p_J(Y_{\text{demo}})p_g(Y_{\text{demo}}). \end{cases} \tag{12}$$

When samples have a high model-likelihood p_0, we ignore the demonstration and sample from the model. Otherwise, we trust the demonstration. With the demonstration-augmented target distribution, we modify the way to calculate $\bar{Y}^{(0)}$ in Eq. (10d) as follows to obtain the score estimate:

$$\bar{Y}^{(0)} = \frac{\sum_{Y^{(0)} \in \mathcal{Y}_d^{(i)}} Y^{(0)} w(Y^{(0)})}{\sum_{Y^{(0)} \in \mathcal{Y}_d^{(i)}} w(Y^{(0)})}, \quad w(Y^{(0)}) = \max \left\{ \begin{array}{c} p_d(Y^{(0)})p_J(Y^{(0)})p_g(Y^{(0)}), \\ p_{\text{demo}}(Y^{(0)})p_J(Y_{\text{demo}})p_g(Y_{\text{demo}}) \end{array} \right\}. \tag{13}$$

5 Experimental Results

The experimental section will focus on demonstrating the capabilities of MBD in: (1) its effectiveness as a zeroth-order solver for high-dimensional, non-convex, and non-smooth trajectory optimization problems, and (2) its flexibility in utilizing dynamically infeasible data to enhance performance and

[3] A comparison between the demonstration-augmented MBD and the vanilla MBD is illustrated in Fig. 6 with detailed breakdowns in Appendix A.3.

regularize solutions. Our benchmark shows that MBD outperforms PPO by 59% in various control tasks with 10% computational time.

Beyond control problems, in Appendix A.2, we also show that MBD significantly improves sampling efficiency by an average of 23% over leading baselines in high-dimensional (up to 800d) black-box optimization testbeds [18, 14, 48, 35, 34, 36]. We also apply MBD to optimize an MLP network with 28K parameters in a *gradient-free* manner, achieving 86% accuracy within 2s for the MNIST classification task [2], which is comparable to the gradient-based optimizer (SGD with momentum, 93% accuracy).

Task	CMA-ES	CEM	MPPI	RL	MBD
Hopper	1.12 ± 0.10	0.65 ± 0.12	0.91 ± 0.15	1.40 ± 0.04	$\mathbf{1.53 \pm 0.03}$
Half Cheetah	0.44 ± 0.10	0.22 ± 0.15	0.20 ± 0.14	1.59 ± 0.05	$\mathbf{2.31 \pm 0.19}$
Ant	1.18 ± 0.52	0.85 ± 0.17	0.33 ± 0.45	3.26 ± 1.61	$\mathbf{3.80 \pm 0.35}$
Walker2D	0.83 ± 0.04	1.06 ± 0.04	0.90 ± 0.05	1.09 ± 0.28	$\mathbf{2.63 \pm 0.23}$
Humanoid Standup	0.58 ± 0.01	0.47 ± 0.01	0.53 ± 0.05	0.83 ± 0.02	$\mathbf{0.99 \pm 0.07}$
Humanoid Running	0.60 ± 0.11	0.41 ± 0.16	0.59 ± 0.14	1.80 ± 0.03	$\mathbf{2.92 \pm 0.26}$
Push T	0.39 ± 0.07	0.25 ± 0.09	-0.13 ± 0.09	-0.63 ± 0.16	$\mathbf{0.67 \pm 0.10}$

Table 2: Reward of different methods on non-continuous tasks.

Task	CMA-ES	CEM	MPPI	RL	MBD
Hopper	29.3 s	26.5 s	26.4 s	17 m 45.63 s	26.5 s
Half Cheetah	29.5 s	26.4 s	26.7 s	4 m 18.8 s	26.8 s
Ant	18.4 s	16.1 s	16.0 s	2 m 46.8 s	16.2 s
Walker2D	37.5 s	34.5 s	34.7 s	5 m 1.5 s	34.6 s
Humanoid Standup	20.8 s	17.6 s	17.7 s	4 m 29 s	17.7 s
Humanoid Running	30.8 s	29.7 s	29.6 s	3 m 34.7 s	30.0 s
Push T	10 m 40.0 s	10 m 32.0 s	10 m 32.3 s	67 m 25.6 s	10 m 32.8 s

Table 3: Computational time of different methods on non-continuous tasks.

5.1 MBD for Planning in Contact-rich Tasks

To test the effectiveness of MBD as a trajectory optimizer for systems involving non-smooth dynamics, we run MBD on both locomotion and manipulation tasks detailed in Appendix A.4.1. The locomotion tasks includes hopper, half-cheetah, ant, walker2d, humanoid-standup, and humanoid-running. The selected manipulation task, pushT [13], presents its own challenges due to the complexity introduced by contact dynamics and the long-horizon nature of the task. These tasks are widely considered difficult due to their hybrid nature and high dimensionality.

MBD is compared with the state-of-the-art zeroth-order optimization methods, including CMA-ES [6], CEM [10], and MPPI [50], as well as reinforcement learning (RL) algorithms (e.g., PPO [39] and SAC [17]) on these tasks. Please note that we use MBD and zeroth-order baselines to generate control sequences and replay them in an open-loop manner, whereas RL generates a closed-loop policy. The RL implementation follows the high-performance parallelized framework from Google Brax [16] elaborated in Appendix A.4.3. For the zeroth-order optimizer, we match the iteration and sample number with the MBD. All the experiments were conducted on a single NVIDIA RTX 4070 Ti GPU. Quantitative metrics including the average step reward and the computational time tested over 50 steps repeated for 8 seeds are reported in Tables 2 and 3. MBD substantially outperforms zeroth-order optimization methods and even outperforms RL in most tasks. Specifically, for the pushT task, MBD achieves a significantly higher reward than the RL algorithm thanks to its iterative refinement process, which effectively explores the full control space while keeping fine-grained control to precisely push the object. Compared with the computationally heavy RL algorithms, MBD only requires one-tenth of time, which is similar to other zeroth-order optimization methods. The optimization process of MBD is visualized in Fig. 3, where the iterative refinement process with the model plays a crucial role in optimizing high-dimensional tasks.

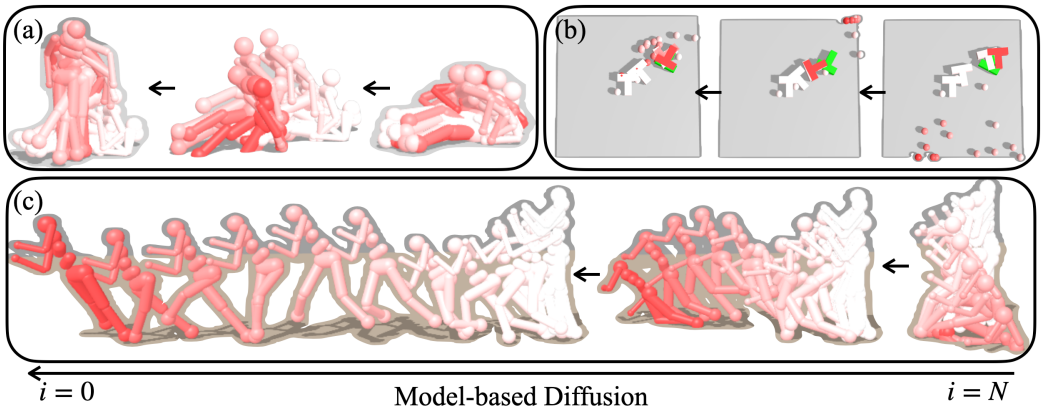

Figure 3: Optimization process of MBD on the (a) Humanoid Standup, (b) Push T, and (c) Humanoid Running tasks. The trajectory is iteratively refined to achieve the desired objective in the high-dimensional space with model information.

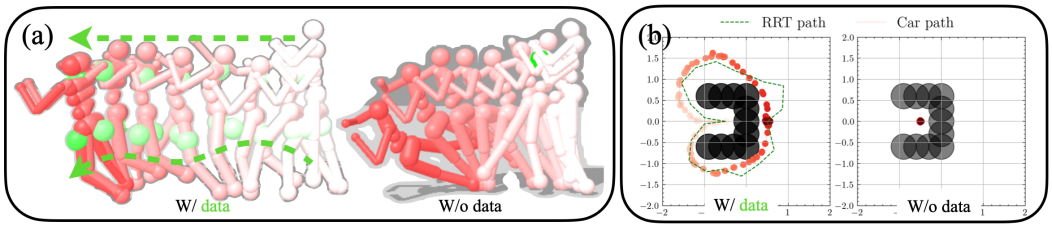

Figure 4: MBD optimized trajectory with data augmentation on the (a) Humanoid Jogging and (b) Car UMaze Navigation tasks. With data augmentation, the trajectory is regularized and refined to achieve the desired objective.

5.2 Data-augmented MBD for Trajectory Optimization

We also evaluate the performance of MBD with data augmentation on the Car UMaze Navigation and Humanoid Jogging tasks to see how partial and dynamically infeasible data can help the exploration of MBD and regularize the solution by steering the diffusion process.

For Car UMaze Navigation, the map blocked by U-shaped obstacles is challenging to explore given a nonlinear dynamics model. Therefore, random shooting has a low chance of reaching the goal region. To sample with loosened dynamical constraints, we augment MBD with data from the RRT [31] algorithm through goal-directed exploration with simplified dynamics. Fig. 4(b) shows the difference between data-augmented MBD and data-free one: the former can refine the infeasible trajectory and further improve it to reach the goal in less time, while the latter struggles to find a feasible solution. The reason is that the infeasible trajectory from RRT serves as a good initialization for MBD, which can be further refined to minimize the cost function with MBD.

For Humanoid Jogging, we aim to regularize the solution for the task with multiple solutions to the desired one with partial state data. Due to the infinite possibilities for humanoid jogging motion, the human motion data provide a good reference to regularize MBD to converge to a more human-like and robust solution instead of an aggressive or unstable one [21, 37]. We use data from the CMU Mocap dataset [1], from which we extract torso, thigh, and shin positions and use them as a partial state reference. Fig. 4(a) demonstrates a more stable motion generated by data-augmented MBD.

6 Conclusion and Future Work

This paper introduces Model-Based Diffusion (MBD), a novel diffusion-based trajectory optimization framework that employs a dynamics model to approximate the score function. MBD not only outperforms existing methods in terms of sample efficiency and generalization, but also provides a new perspective on trajectory optimization by leveraging diffusion models as powerful samplers. Future directions involve theoretically understanding its convergence, optimizing the standard Gaussian for-

ward process using the model information, adapting it to online tasks with receding horizon strategies, and exploring advanced sampling and scheduling techniques to further improve performance.

References

[1] Carnegie Mellon University - CMU Graphics Lab - motion capture library. http://mocap.cs.cmu.edu/.

[2] The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web] | IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/6296535.

[3] Sequential Quadratic Programming. In Jorge Nocedal and Stephen J. Wright, editors, *Numerical Optimization*, pages 529–562. Springer, New York, NY, 2006.

[4] David H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*, volume 28 of *The Kluwer International Series in Engineering and Computer Science*. Springer US, Boston, MA, 1987.

[5] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is Conditional Generative Modeling all you need for Decision-Making?, July 2023.

[6] Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Theoretical Foundation for CMA-ES from Information Geometry Perspective. *Algorithmica*, 64(4):698–716, December 2012.

[7] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces, February 2023.

[8] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization, December 2020.

[9] Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling, July 2023.

[10] Zdravko I. Botev, Dirk P. Kroese, Reuven Y. Rubinstein, and Pierre L'Ecuyer. Chapter 3 - The Cross-Entropy Method for Optimization. In C. R. Rao and Venu Govindaraju, editors, *Handbook of Statistics*, volume 31 of *Handbook of Statistics*, pages 35–59. Elsevier, January 2013.

[11] João Carvalho, An T. Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion Planning Diffusion: Learning and Planning of Robot Motions with Diffusion Models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923, Detroit, MI, USA, October 2023. IEEE.

[12] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan. WaveGrad: Estimating Gradients for Waveform Generation, October 2020.

[13] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion, June 2023.

[14] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[15] Peter I. Frazier. A Tutorial on Bayesian Optimization, July 2018.

[16] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – A Differentiable Physics Engine for Large Scale Rigid Body Simulation, June 2021.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[18] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial, March 2023.

[19] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, March 2003.

[20] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, May 2012.

[21] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning Human-to-Humanoid Real-Time Whole-Body Teleoperation, March 2024.

[22] José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán 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*, pages 1470–1479. PMLR, July 2017.

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, December 2020.

[24] Taylor A. Howell, Brian E. Jackson, and Zachary Manchester. ALTRO: A Fast Solver for Constrained Trajectory Optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7674–7679, Macau, China, November 2019. IEEE.

[25] Wilson Jallet, Antoine Bambade, Nicolas Mansard, and Justin Carpentier. Constrained Differential Dynamic Programming: A primal-dual augmented Lagrangian approach, October 2022.

[26] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis, December 2022.

[27] Chiyu Max Jiang, Andre Cornman, Cheolho Park, Ben Sapp, Yin Zhou, and Dragomir Anguelov. MotionDiffuser: Controllable Multi-Agent Motion Prediction using Diffusion, June 2023.

[28] B. Kappen, V. Gomez, and M. Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, May 2012.

[29] Seung-Jean Kim, K. Koh, M. Lustig, Stephen Boyd, and Dimitry Gorinevsky. An Interior-Point Method for Large-Scale -Regularized Least Squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, December 2007.

[30] Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, June 2012.

[31] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[32] Sergey Levine. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review, May 2018.

[33] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. AdaptDiffuser: Diffusion Models as Adaptive Self-evolving Planners, May 2023.

[34] Jialin Liu, Antoine Moreau, Mike Preuss, Baptiste Roziere, Jeremy Rapin, Fabien Teytaud, and Olivier Teytaud. Versatile Black-Box Optimization, April 2020.

[35] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A Framework for Bayesian Optimization in Embedded Subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4752–4761. PMLR, May 2019.

[36] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces, April 2023.

[37] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Transactions on Graphics*, 40(4):1–20, August 2021.

[38] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-Conditioned Imitation Learning using Score-based Diffusion Policies, June 2023.

[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

[40] Guanya Shi, Wolfgang Honig, Xichen Shi, Yisong Yue, and Soon-Jo Chung. Neural-Swarm2: Planning and Control of Heterogeneous Multirotor Swarms Using Learned Interactions. *IEEE Transactions on Robotics*, 38(2):1063–1079, April 2022.

[41] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[42] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, November 2015.

[43] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, October 2020.

[44] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution, October 2020.

[45] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, February 2021.

[46] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human Motion Diffusion Model, October 2022.

[47] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1049–1056, Montreal Quebec Canada, June 2009. ACM.

[48] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search. In *Advances in Neural Information Processing Systems*, volume 33, pages 19511–19522. Curran Associates, Inc., 2020.

[49] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model Predictive Path Integral Control: From Theory to Parallel Computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, February 2017.

[50] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, December 2018.

[51] Wei Xiao, Tsun-Hsuan Wang, Chuang Gan, and Daniela Rus. SafeDiffuser: Safe Planning with Diffusion Probabilistic Models, May 2023.

[52] Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional Diffusion-Based Continuous Constraint Solvers, September 2023.

[53] Zeji Yi, Chaoyi Pan, Guanqi He, Guannan Qu, and Guanya Shi. CoVO-MPC: Theoretical Analysis of Sampling-based MPC and Optimal Covariance Design, January 2024.

[54] Zeji Yi, Yunyue Wei, Chu Xin Cheng, Kaibo He, and Yanan Sui. Improving sample efficiency of high dimensional Bayesian optimization with MCMC, January 2024.

[55] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method, December 2012.

A Appendix / Supplemental Material

A.1 Convergence of Distribution with Small λ

We first give the definition of the volume of the sub-level set for cost J.

Definition 1. *Let $F : \mathcal{R}^d \to \mathcal{R}$ be a measurable function. Define the volume of the sub-level set for a given level t as:*

$$V_F(t) = \int_{\mathcal{R}^d} \chi_{\{Y \in \mathcal{R}^d : F(Y) \le t\}}(Y)\, dY,$$

where χ denotes the indicator function.

The volume function $V_J(t)$ plays a crucial role in linking geometric properties with probabilistic outcomes in optimization and learning algorithms. This function provides a quantitative measure that helps us to understand how changes in parameters like λ influence the distribution and concentration of probability mass.

The interplay between geometry and probability, represented by $V_J(t)$, is crucial for evaluating the convergence and stability of algorithms. It provides a significant method for utilizing the PDF of the random variable $\mathbf{Y}$ to constrain the CDF, thereby facilitating convergence in distribution.

Proposition 2. *Given the target distribution $\mathbf{Y} \sim p(\cdot)$ with $P(Y) \propto \exp\left(-\frac{J(Y)}{\lambda}\right), Y \in \mathcal{R}^d$, where J is a cost function with $\min_Y J(Y) = 0$ and $Y^* = \arg\min J(Y)$, and assuming that the volume function $V_J(t)$ is bounded by polynomial inequalities:*

$$Poly_l(t) \le V_J(t) \le Poly_u(t),$$

where $Poly_l(t) = \sum_{k=0}^{M} c_k^l t^{\alpha_k}$ and $Poly_u(t) = \sum_{k=0}^{M} c_k^u t^{\alpha_k}$ are polynomials with coefficients satisfying $c_k^l = 0$ if and only if $c_k^u = 0$. The exponent term satisfies that $\alpha_k \in \mathcal{R}$, and $0 < \alpha_0 < \alpha_1 < \cdots < \alpha_M < \infty$, It follows that:

$$\lim_{\lambda \to 0} J(\mathbf{Y}) \xrightarrow{p} J(Y^*) = 0.$$

The cost value $J(Y)$ converges in probability to $J(Y^*)$ as $\lambda \to 0$.

The condition on the polynomial bounds of $V_J(t)$ is generally not restrictive. For instance, consider $J = \eta_c \left\| Y - Y_* \right\|^m$, where Y^* is the optimal point and $\eta_c > 0$ is any constant. In this case, $V_J(t) = Ct^{\frac{d}{m}}$, where C is a constant, meets the constraint in a straightforward way. This condition can be extended beyond this simple scenario, as even if J has multiple modes, it can still adhere to this polynomial constraint.

Proof. The convergence in distribution of $\boldsymbol{Y}$ towards Y^* as $\lambda \to 0$ is established by analyzing the behavior of the probability density function, defined up to a multiplicative constant. Consider the density $\boldsymbol{Y}_0 \sim p_\lambda(Y)$ approximating Y^* when λ approaches zero.

$$P(J(\mathbf{Y}) \le t) = \int_{\{J(Y) \le t\}} p(Y) dY, \tag{14a}$$

$$= \int_0^t \int_{\{J(Y)=x\}} p(Y) dY dx, \tag{14b}$$

$$\propto \int_0^t exp(-\frac{x}{\lambda}) \int_{\{J(Y)=x\}} dY dx, \tag{14c}$$

$$= \int_0^t exp(-\frac{x}{\lambda}) \frac{dV_J(x)}{dx} dx, \tag{14d}$$

where Eq. (14c) is valid since $P(Y) \propto \exp\left(-\frac{J(Y)}{\lambda}\right)$ and $J(Y)$ represents the sufficient statistics of the distribution. We can obtain Eq. (14d) by computing the derivative of $V_J(x)$ based on the volume definition as shown in Definition 1.

We denote $J_{\min} = \min_Y J(Y) = 0$ and $J_{\max} = \max_Y J(Y)$ with $J_{\max}$ satisfying $0 \le J_{\max} \le +\infty$. We proceed to analyze Eq. (14d) by performing integration by parts as shown in Eq. (15a).

$$\int_{J_{\min}}^{J_{\max}} \exp\left(-\frac{t}{\lambda}\right) dV_J(t) = \int_{J_{\min}}^{J_{\max}} d\left[\exp\left(-\frac{t}{\lambda}\right) V_J(t)\right] + \frac{1}{\lambda} \exp\left(-\frac{t}{\lambda}\right) V_J(t) dt \tag{15a}$$

$$= \exp\left(-\frac{t}{\lambda}\right) V_J(t) \Big|_{J_{\min}}^{J_{\max}} + \frac{1}{\lambda} \int_{J_{\min}}^{J_{\max}} \exp\left(-\frac{t}{\lambda}\right) V_J(t) dt. \tag{15b}$$

To establish convergence in probability, we need to demonstrate that for any small $\epsilon > 0$ and $\delta > 0$, there exists sufficiently small $\lambda > 0$, such that

$$P(J(\boldsymbol{Y}) < \epsilon) = \frac{\int_0^\epsilon \exp\left(-\frac{t}{\lambda}\right) dV_J(t)}{\int_0^{J_{\max}} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)} \ge 1 - \delta. \tag{16}$$

where the equality is due to Eq. (14d). Setting $\delta' = \frac{1-\delta}{\delta}$, it suffices to show that:

$$\frac{\int_0^{\epsilon} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)}{\int_{\epsilon}^{J_{\max}} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)} \geq \delta'. \tag{17}$$

Assuming without loss of generality that $J_{\max} = \infty$, becuase $dV_J(t) \geq 0$, $exp(-\frac{t}{\lambda}) > 0$, we have:

$$\frac{\int_0^{\epsilon} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)}{\int_{\epsilon}^{J_{\max}} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)} \geq \frac{\int_0^{\epsilon} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)}{\int_{\epsilon}^{\infty} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)}. \tag{18}$$

This ratio as in Eq. (17) can be expanded using the integral bounds and the polynomial approximations for $V_J(t)$,then it suffices to show that

$$\frac{\int_0^{\epsilon} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)}{\int_{\epsilon}^{\infty} \exp\left(-\frac{t}{\lambda}\right) dV_J(t)} \geq \delta'. \tag{19}$$

By inserting Eq. (15b) into both the numerator and denominator on the LHS of Eq. (19), we obtain

$$\frac{\int_0^{\epsilon} exp(-\frac{t}{\lambda}) dV_J(t)}{\int_{\epsilon}^{\infty} exp(-\frac{t}{\lambda}) dV_J(t)} = \frac{exp(-\frac{\epsilon}{\lambda})V(\epsilon) + \frac{1}{\lambda}\int_0^{\epsilon} exp(-\frac{t}{\lambda})V_J(t)dt}{-exp(-\frac{\epsilon}{\lambda})V(\epsilon) + \frac{1}{\lambda}\int_{\epsilon}^{\infty} exp(-\frac{t}{\lambda})V_J(t)dt} \tag{20a}$$

$$\geq \frac{\int_0^{\epsilon} exp(-\frac{t}{\lambda})V_J(t)dt}{\int_{\epsilon}^{\infty} exp(-\frac{t}{\lambda})V_J(t)dt} \tag{20b}$$

$$\geq \frac{\int_0^{\epsilon} exp(-\frac{t}{\lambda}) \sum_{k=0}^{M} c_k^l t^{\alpha_k} dt}{\int_{\epsilon}^{\infty} exp(-\frac{t}{\lambda}) \sum_{k=0}^{M} c_k^u t^{\alpha_k} dt} \tag{20c}$$

To bound the expression in Eq. (20c), we first derive the following integrals by utilizing a change of variables $x = \frac{t}{\lambda}$, which simplifies the expressions:

$$\int_0^{\epsilon} \exp\left(-\frac{t}{\lambda}\right) t^{\alpha_k}\, dt = \lambda^{k+1} \int_0^{\frac{\epsilon}{\lambda}} \exp(-x) x^{\alpha_k}\, dx, \tag{21a}$$

$$\int_{\frac{\epsilon}{\lambda}}^{\infty} \exp\left(-\frac{t}{\lambda}\right) t^{\alpha_k}\, dt = \lambda^{k+1} \int_{\frac{\epsilon}{\lambda}}^{\infty} \exp(-x) x^{\alpha_k}\, dx. \tag{21b}$$

For these transformed integrals, we can observe that $\int_0^{\infty} \exp(-x) x^{\alpha_k}\, dx = \Gamma(\alpha_k + 1)$, the gamma function, which is well-defined for all non-negative α_k. Given that δ' is a function of δ , by applying the intermediate value theorem and definition of the limit of the integral, we can choose ϵ_k in such a way that:

$$\int_0^{\epsilon_k} \exp(-x) x^{\alpha_k}\, dx \geq \frac{c_k \delta'}{1 + c_k \delta'} \Gamma(\alpha_k + 1),$$

where $c_k = \frac{c_k^l}{c_k^u}$ denotes the ratio of coefficients in polynomial lower and upper bounds for $V_J(t)$. By selecting $\epsilon_{\max} = \max \epsilon_0, \epsilon_1, \cdots, \epsilon_M$ to be the maximum of all such ϵ_k, ensuring coverage for all polynomial terms up to M, we establish that:

$$\frac{\int_0^{\epsilon_{\max}} \exp\left(-\frac{t}{\lambda}\right) c_k^l t^{\alpha_k}\, dt}{\int_{\epsilon_{\max}}^{\infty} \exp\left(-\frac{t}{\lambda}\right) c_k^u t^{\alpha_k}\, dt} \geq \delta', \quad \text{for all } k = 0, 1, \ldots, M. \tag{22}$$

By ensuring that $\lambda \leq \frac{\epsilon}{\epsilon_{\max}}$, we can conclude:

$$\frac{\int_0^{\epsilon} \exp\left(-\frac{t}{\lambda}\right) \sum_{k=0}^{M} c_k^l t^{\alpha_k}\, dt}{\int_{\epsilon}^{\infty} \exp\left(-\frac{t}{\lambda}\right) \sum_{k=0}^{M} c_k^u t^{\alpha_k}\, dt} \geq \delta', \tag{23}$$

Thus, the condition specified in Eq. (16) is satisfied, validating that the distribution of $\boldsymbol{Y}$ converges in distribution to Y^* as λ approaches zero.

□

By adding another mild assumption regarding the landscape of J near the global optimum, we can demonstrate the convergence of the random variable $\mathbf{Y}$ itself, rather than the convergence of $J(\mathbf{Y})$.

Definition 3. *We denote the minimum of the complementary set of neighborhood as:*

$$J^*_{\mathcal{B}}(\delta) = \min_{\|Y-Y^*\|>\delta} J(Y) - J(Y^*).$$

Proposition 4. *Given the context and conditions specified in Definitions 1 and 3 and Proposition 2, and given that J has only one golbal minimizer Y^*, i.e. there exist small δ^*, that for $\delta \in (0, \delta^*]$, $J^*_{\mathcal{B}}(\delta)$ is strictly increasing, and $J^*_{\mathcal{B}}(\delta^*) < \infty$. It follows that:*

$$\lim_{\lambda\to 0} \boldsymbol{Y} \xrightarrow{p} Y^*.$$

The random variable $\mathbf{Y}$ converges in probability to Y^* as $\lambda \to 0$.

Proof. In order to prove that $\lim_{\lambda\to 0} \boldsymbol{Y} \xrightarrow{p} Y^*$. We need to prove that for any sufficient small $\gamma > 0$ and $\delta > 0$, there exists small $\lambda > 0$, such that

$$P(\|Y - Y^*\| \le \delta) \ge 1 - \gamma \tag{24}$$

From Definition 3 and due to the strict increase of $J^*_{\mathcal{B}}(\delta)$,

$$\|Y - Y^*\| \le \delta, \quad \forall Y \in \left\{Y \in \mathcal{R}^d \mid J(Y) - J(Y^*) < J^*_{\mathcal{B}}(\delta)\right\}, \tag{25}$$

where $0 < \delta \le \delta^*$. Because if $\|Y - Y^*\| > \delta$, $J(Y) - J(Y^*) < J^*_{\mathcal{B}}(\delta) = \min_{\|Y-Y^*\|>\delta} J(Y) - J(Y^*)$ contradicts Definition 3.

Given that $\lim_{\lambda\to 0} J(\boldsymbol{Y}) \xrightarrow{p} J(Y^*)$. and any sufficient small $\epsilon, \gamma > 0$.

$$P(J(Y) - J(Y^*) \le \epsilon) \ge 1 - \gamma \tag{26}$$

Therefore, $\exists\lambda > 0$, such that

$$P(J(Y) - J(Y^*) \le J^*_{\mathcal{B}}(\delta)) \ge 1 - \gamma. \tag{27}$$

And From Eq. (25), we have that

$$P(\|Y - Y^*\| \le \delta) \ge P(J(Y) - J(Y^*) \le J^*_{\mathcal{B}}(\delta)) \ge 1 - \gamma \tag{28}$$

We have that $\mathbf{Y}$ converges in probability to Y^* ,i.e, $\lim_{\lambda\to 0} \boldsymbol{Y} \xrightarrow{p} Y^*$.

□

Proposition 5. *Given the context and conditions specified in Propositions 2 and 4 and the way we define the forward process as in Eq.* (3)*. The diffused Y_i converge in density to a Gaussian distribution.*

$$\lim_{\lambda\to 0} \boldsymbol{Y^{(i)}} \xrightarrow{d} \mathcal{N}(\sqrt{\bar{\alpha}_i}Y^*, \sqrt{1-\bar{\alpha}_i}I),$$

where $\boldsymbol{Y^{(i)}} \sim p_i(\cdot)$ as in Eq. (3)*.*

Proposition 5 is derived by using Slutsky's theorem on Proposition 4 and offers insight into choosing the stepsize as discussed in Section 4.1.

A.2 Black-box Optimization with MBD

As a zeroth order optimizer, MBD is capable of addressing both trajectory optimization and broader, high-dimensional unconstrained optimization challenges. Such black-box optimization tasks are universally acknowledged as difficult [6, 54]. We first show superior performance of MBD within this black-box optimization context. In such settings, the Bayesian Optimization technique struggles due to the computational intensity required to develop surrogate models and identify new potential solutions [14]. Alternative black-box optimization strategies [18] are not limited by computational issues but tend to be less efficient because they do not estimate the black-box function as accurately. MBD's effectiveness is evaluated using two well-known highly non-convex black-box optimization benchmarks: Ackley [4] and Rastrigin [8], each tested across three different dimensionalities. Comparisons were made with CMA-ES [18], TuRBO [14], LA-MCTS [48], HesBO [35], Shiwa [34], and BAxUS [36].

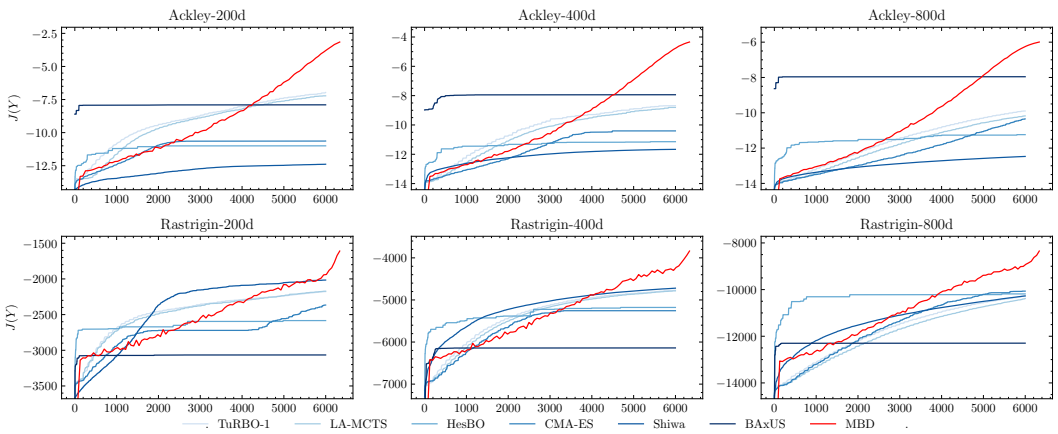

Figure 5: Performance of MBD on high-dimensional black-box optimization benchmarks. MBD outperforms other Gaussian Process-based Bayesian Optimization methods by a clear margin.

Fig. 5 shows the performance of MBD on the Ackley and Rastrigin benchmarks. MBD demonstrates superior performance over other algorithms for several reasons. Firstly, the implementation of a scheduled forward process that determines the total number of samples consequently boosts sample efficiency. Secondly, the application of Monte Carlo score ascent on various $\log p_i(Y^{(i)})$ facilitates its escape from local optima of varying scales. It is important to acknowledge that comparing computational efficiency may not be entirely fair, given that black-box optimization problems typically involve functions that are costly to evaluate. However, MBD markedly outperforms other Gaussian Process-based Bayesian Optimization approaches, achieving computational time savings of more than twentyfold, similar to the improvements observed with different evolutionary optimization strategies.

Here are the implementation detail for the benchmarks. For the BO benchmarks, the experiments were conducted on an A100 GPU because of the high computational demands of the Gaussian Process Regression Model it incorporates.

TuRBO: TuRBO is implemented based on tutorials from Botorch [8].

LA-MCTS: LA-MCTS, we refer to authors' reference implementations, and use TuRBO as its local BO solver [48].

HesBO: For HesBO, we refer to authors' reference implementations [35]. We transformed default GP component into Gpytorch version for faster inference speed on GPU. We set the embedding dimension to 20 for all tasks

CMA-ES: We use pycma[4] to implement CMA-ES, and use default setting except setting population size eqauls to batch size.

Shiwa: We use Nevergrad[5] to implement Shiwa, and use default setting to run experiments.

BAxUS: We refer to the authors' reference implementations [36].

A.2.1 MBD for DNN Training without Gradient Information

To further demonstrate the effectiveness of MBD in high-dimensional systems, we apply MBD to optimize an MLP network for MNIST classification [2] without access to the gradient information. MBD achieve 85.5% accuracy with 256 samples within 2s, which is comparable to the performance of the SGD optimizer with momentum (92.7% accuracy). We use MLP with 2 hidden layers, each with 32 neurons, and ReLU activation function. The input is flattened to 784 dimensions, and the output is a 10-dimensional vector. We use cross-entropy loss as the objective function. The network has $27,562$ parameters in total, which makes sampling-based optimization challenging. MBD can effectively optimize the network with a small number of samples, demonstrating its effectiveness in high-dimensional black-box optimization tasks.

[4]https://github.com/CMA-ES/pycma

[5]https://github.com/facebookresearch/nevergrad

A.3 MBD with Demonstration Explaination

Data-augmented MBD calculate the score function with demostration as follows:

$$Y^{(i-1)} = \frac{1}{\sqrt{\alpha_i}}\left(Y^{(i)} + (1-\bar{\alpha}_i)\nabla_{Y^{(i)}} \log p_i(Y^{(i)})\right) \tag{29}$$

$$\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) = -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i}\bar{Y}^{(0)} \tag{30}$$

$$\text{where} \quad \bar{Y}^{(0)} = \frac{\sum_{Y^{(0)}\in\mathcal{Y}_d^{(i)}} Y^{(0)} w(Y^{(0)})}{\sum_{Y^{(0)}\in\mathcal{Y}_d^{(i)}} w(Y^{(0)})} \tag{31}$$

$$w(Y^{(0)}) = \max\left\{ \begin{array}{c} w_{\text{model}}(Y^{(0)}) = p_d(Y^{(0)})p_J(Y^{(0)})p_g(Y^{(0)}), \\ w_{\text{demo}}((Y^{(0)})) = p_{\text{demo}}(Y^{(0)})p_J(Y_{\text{demo}})p_g(Y_{\text{demo}}) \end{array} \right\} \tag{32}$$

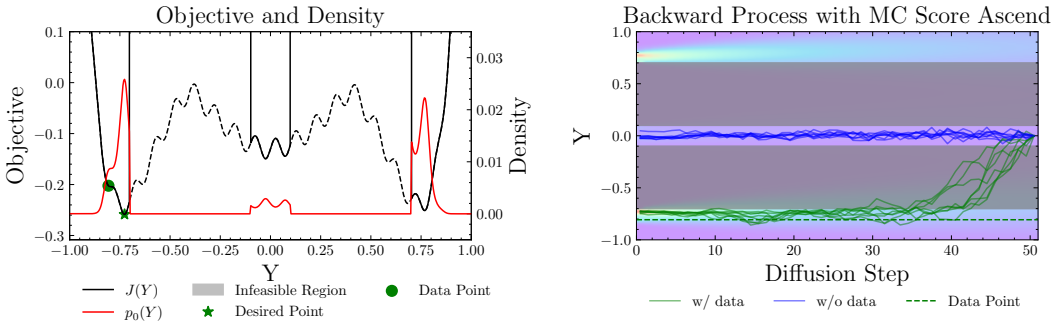

Figure 6: MBD with data vs. without data on a nonconvex function with constraints $||Y| - 0.4| > 0.3$. We want MBD converge to the optimal point ★ with the help of demonstration data ●. Although the demostration point is not optimal, MBD can still converge to the optimal point with the guidance of the demonstration data. Here data serves as a regularization term to guide the diffusion process to the negative optimal point while allowing to use model further to refine the solution.

where demonstrate likelihood term $w_{\text{demo}}(Y^{(0)})$ will draw samples towards data without considering the model. Given $w = w_{\text{demo}}$, $\bar{Y}^{(0)} = \frac{\sum_{Y^{(0)}\in\mathcal{Y}_d^{(i)}} Y^{(0)} p_{\text{demo}}(Y^{(0)})}{\sum_{Y^{(0)}\in\mathcal{Y}_d^{(i)}} p_{\text{demo}}(Y^{(0)})} = Y_{\text{demo}}$. The score function would be $\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) = -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} Y_{\text{demo}}$, which means the score function is a linear combination of the current sample and the demonstration data.

If we don't use Eq. (11) and employ the posterior distribution $p(Y^{(0)}|Y_{\text{demo}}) \propto p_0(Y^{(0)})p_{\text{demo}}(Y^{(0)})$, it will yields update weights $w = w_{\text{demo}} w_{\text{model}}$, which will draw samples to both model and demonstration data. If the demonstration data is not optimal, the final solution will be a compromise between the model and demonstration data. In Fig. 6, the resulted solution will lie between optimal point ★ with the help of demonstration data ●.

Using the max function in w can aviod this issue. In the early stage while $p_J(Y^{(0)})$ is low due to poor sample quality, w_{demo} will dominate thanks to the high $p_J(Y_{\text{demo}})$. This will draw samples towards the demonstration data as shown in the earlier stage of Fig. 6. As the sample quality improves and $p_J(Y^{(0)}) > p_J(Y_{\text{demo}})$, w_{model} will dominate and the sample will converge to the optimal point.

A.4 Experiment Details

A.4.1 Simulator and Environment

We leverage the GPU-accelerated simulator Google Brax [16] to design the locomotion and manipulation tasks. All task is set to use positional backend in Brax except for the pushT task, which uses the generalizable backend for better contact dynamics simulation. Here we provide a brief description of each task implementations:

1. **Ant**: The Ant task is a 3D locomotion task where the agent is required to move forward as fast as possible. The reward is composed of the forward velocity of the agent and control cost, same as the original Brax implementation. The control dimension is 8.
2. **Hopper**: The Hopper task is a 2D locomotion task where the agent is required to jumping forward as fast as possible. We use the same reward function as the original Brax implementation. We modify the simulation substeps from 10 to 20 for longer planning horizon given the same control node. The control dimension is 3.
3. **Walker2d**: The Walker2d task is a 2D locomotion task where the agent is required to walk forward. The reward is composed of keep the agent upright and moving forward. The control dimension is 6.
4. **Halfcheetah**: The Halfcheetah task is a 2D locomotion task where the agent is required to run forward. The reward is composed of the forward velocity of the agent and control cost. We follow the same reward function as the original Brax implementation. The control dimension is 6.
5. **Humanoidrun**: The Humanoidrun task is a 3D locomotion task where the agent is required to run forward. The reward is composed of the forward velocity of the agent and standing upright. Here we also modify the simulation substeps from 10 to 20 for longer planning horizon. The control dimension is 17.
6. **Humanoidstandup**: The Humanoidstandup task is a 3D locomotion task where the agent is required to stand up. The reward is the upright torso position of the agent. The control dimension is 17.
7. **PushT**: The PushT task is a 2D manipulation task where you can apply force to a sphere to push the T-shaped object to the target location. The reward is composed of the distance between the target and the object and orientation difference between the target and the object. To make the task more challenging, we randomize the target location 20cm away from the initial position and make sure the rotational angle is greater than 135 degrees, which makes it hard to solve the task with single continous contact policy. The control dimension is 2.
8. **Car2D**: We implement a 2D car task with standard bicycle dynamics model, where state is $x = [x, y, \theta, v, \delta]$, and action is $u = [a, \delta]$. The dynamics is defined as $\dot{x} = f(x, u) = [v\cos(\theta), v\sin(\theta), \frac{v}{L}\tan(\delta), a, \delta]$. The constraints are defined as the U-shape area in the middle of the map, where the car cannot enter. The reward is composed of the distance between the target and the car and the control cost. The control dimension is 2.

A.4.2 MBD Hyperparameters

In general, MBD is very little hyperparameters to tune compared with RL. We use the same hyperparameters for all the tasks, with small tweaks for harder tasks.

Task Name	Horizon	Sample Number	Temperature λ
Ant	50	100	0.1
Halfcheetah	50	100	0.4
Hopper	50	100	0.1
Humanoidstandup	50	100	0.1
Humanoidrun	50	300	0.1
Walker2d	50	100	0.1
PushT	40	200	0.2

Table 4: MBD hyperparameters for various tasks

For diffusion noise schedulling, we use simple linear scheduling $\beta_0 = 1 \times 10^{-4}$ and $\beta_N = 1 \times 10^{-2}$, and the diffusion step number is 100 across all tasks. Each step's α_i is calculated as $\alpha_i = 1 - \beta_i$.

A.4.3 Baseline Algorithms Implementation

For reinforcement learning implementation, we strictly follow the hyperparameters and implementation details provided by the original Brax repository, which optimize for the best performance. For our self-implemented PushT task, the hyperparameters is ported from Pusher task in Brax for fair comparison. The hyperparameters for the RL tasks are shown in Table 5 and Table 6.

For the zeroth order optimization tasks, we the same hyperparameters as the MBD algorithm.

Environment	Algorithm	Timesteps	Reward Scaling	Episode Length
Ant	PPO	100M	10	1000
Hopper	SAC	6.55M	30	1000
Walker2d	PPO	50M	1	1000
Halfcheetah	PPO	50M	1	1000
Pusher	PPO	50M	5	1000
PushT	PPO	100M	1.0	100
Humanoidrun	PPO	100M	0.1	100
Humanoidstandup	PPO	100M	0.1	1000

Table 5: General RL configuration for various environments

Environment	Minibatches	Updates/Batch	Discounting	Learning Rate
Ant	32	4	0.97	3×10^{-4}
Hopper	32	4	0.997	6×10^{-4}
Walker2d	32	8	0.95	3×10^{-4}
Halfcheetah	32	8	0.95	3×10^{-4}
Pusher	16	8	0.95	3×10^{-4}
PushT	16	8	0.99	3×10^{-4}
Humanoidrun	32	8	0.97	3×10^{-4}
Humanoidstandup	32	8	0.97	6×10^{-4}

Table 6: RL specifics for various environments

A.4.4 Demonstration Collections

For RRT algorithm in Car2D task, we set the max step size to 0.2, and the max iterations to 1000 given the maximum episode length is 50.

For the demonstration collection in Humanoid Jogging task, we first download the mocap data which contains each joints' position in the world frame. Then we use the joint data to calculate the position of torso, thigh and shin position as partial state reference for our task.