IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Importance Sampling Model-Based Diffusion for Trajectory Optimization

Seth Golembeski[1] and Anirban Mazumdar[2]

***Abstract*—Trajectory optimization for robotic systems remains a challenging problem. This is especially true for robotic systems featuring nonlinear dynamics and many degrees of freedom. Data-based or model-free diffusion has recently been popularized in the fields of artificial intelligence and trajectory optimization. Model-Based Diffusion provides a data-free method of trajectory optimization, trained at runtime on a system dynamics model, suitable for high-dimensional models. This paper examines how importance sampling can enhance the performance of Model-Based Diffusion for trajectory optimization. We quantify the benefits of importance sampling across three long horizon planning tasks. These results show as much as a 13x improvement in sample efficiency depending on environment and optimization parameters.**

***Index Terms*—Motion and Path Planning, Optimization and Optimal Control, Nonholonomic Motion Planning**

I. INTRODUCTION

TRAJECTORY Optimization (TO) methods are used across robotics to minimize a cost function $J(u, x)$ via optimization of control inputs u. However, many conventional TO methods have limited efficacy in problems which are non-convex and non-linear. For example, direct collocation requires C_2 functions [1] and gradient-based methods like shooting are subject to both this continuity constraint and sensitivity to initial conditions [2].

Diffusion models have recently been popularized for use in generative image [3], text [4], and video [5] creation. These tasks have similarities to TO. Specifically, both generative image/video creation and TO are complex optimization problems operating on time-series or structured data. In addition, the cost functions are often computationally expensive, non-linear, and non-convex. Model-free diffusion methods have been developed to apply diffusion to trajectory optimization. However, they are largely data-based, and require a training dataset or reduced-order model prior to use [6, 7, 8, 9]. Model-based diffusion (MBD), in contrast, uses a dynamic model that is applied at runtime, eliminating the need for a training dataset or training time [10]. These recent results illustrate the potential for model-based diffusion to improve TO for non-convex and/or non-linear systems. One particularly exciting area is the use of MBD for long-horizon kinodynamic planning tasks, as identified in [10]. However, the number of samples required for problems with long horizons or large input spaces can be prohibitive.

Since sampling is core to the MBD algorithm, we posit that the efficiency of MBD can be greatly improved by performing more intelligent sampling. To this end, we replace the standard Gaussian sampler with a cross-entropy (CE) adaptive importance sampler (AIS).

In this work we present Model Based Diffusion with Adaptive Importance Sampling (MBD-AIS). As we show in Fig. 1, MBD-AIS more effectively shapes MBD sampling, with the quantitative impact demonstrated in VI. This approach significantly reduces the number of samples required for problems with large input spaces, as we demonstrate for cases including a car racing environment given by [11] and a variety of MuJoCo environments. This paper provides the following contributions:

- Performance quantifications of MBD-AIS.
- Demonstration of a significant improvement in sample efficiency, with improvements as large as 13x.
- Benchmarking MBD-AIS against MBD [10], MPPI [12] and MPOPI-CE [11] sampling based optimization, validating the findings of [10], and further verifying MBD-AIS performance in long-horizon tasks.

II. BACKGROUND

A. Trajectory Optimization

MBD [10] operates on the family of TO problems which minimize the cost function $J(\mathbf{x}_t, \mathbf{u}_t)$ subject to dynamics $f(\mathbf{x}_t, \mathbf{u}_t)$ and constraints $g(\mathbf{x}_t, \mathbf{u}_t)$. States $\mathbf{x}_t \in \mathbb{R}^{n_x}$ represent the n_x states of the system at any given time, whereas control inputs $\mathbf{u}_t \in \mathbb{R}^{n_u}$ represent the n_u control inputs of the system.

In this paper, subscript $\cdot_t$ represents time while superscript $\cdot^i$ represents a diffusion step, unless otherwise defined. Time ranges from values $\{0, 1, \ldots, T-1, T\}$ and i ranges from $\{1, 2, \ldots, n_\mathrm{D} - 1, n_\mathrm{D}\}$. The TO problem seeks to find the optimal control sequence, $u^*(t)$. This can be written as

$$u^*(t) = \mathrm{argmin}_u \left(\sum_{t=0}^{T-1} J(\mathbf{x}_t, \mathbf{u}_t) \right), \tag{1}$$

subject to:

$$\begin{cases} \mathbf{x}_0 = \mathbf{x}_{init} \\ \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \\ g(\mathbf{x}_t, \mathbf{u}_t) \leq 0. \end{cases} \tag{2}$$

Manuscript received: May 16, 2025; Revised: August 13, 2025; Accepted: September 16, 2025.

This paper was recommended for publication by Editor Cosimo Della Santina upon evaluation of the Associate Editor and Reviewers' comments.

[1] Seth Golembeski is with the Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: sgolembeski3@gatech.edu).

[2] Anirban Mazumdar is with the George W. Woodruff Schoool of Mechanical Engineering, Atlanta, GA, 30332 USA and Sandia National Laboratories, Albuquerque, NM, 87123 USA (e-mail: anirban.mazumdar@me.gatech.edu).

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

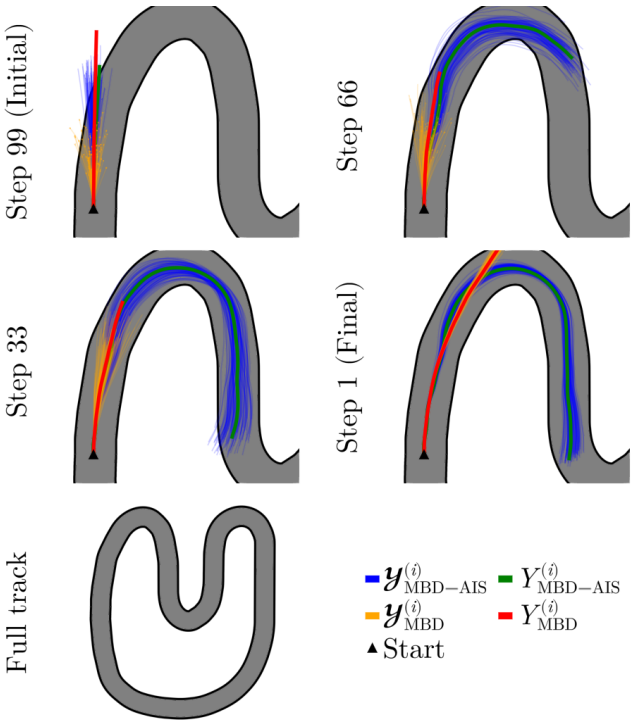

Fig. 1: A figure showing the MBD and MBD-AIS operating on the car racing environment described in VI-A. As the diffusion process progresses, the sampling distribution changes. Note the differing distribution shapes, as well as the faster progress of MBD-AIS.

B. Related TO Methods

Numerous methods exist to solve the above equations. These include sampling-based methods like RRT [13] or optimal RRT [14] and TO methods such as direct optimization/collocation [1] or shooting [2]. All of these families of methods struggle with either high-dimensional states and inputs, non-convex problems, or the nonlinear dynamics which are common in trajectory optimization problems [1, 2, 13, 14, 15].

Machine learning (ML) methods such as genetic algorithms [2], imitation learning [16], data-based diffusion, or Reinforcement Learning (RL) — for example, Proximal Policy Optimization (PPO) [17] — can also be used. However, these methods require a rich input dataset and may be sensitive to changes in system dynamics, input states, or environment states.

C. Diffusion Models

Diffusion models work by initializing the diffusion state $Y^{(i)}$ (initially, $i = n_{\text{D}}$) with noise and iteratively denoising it from $i = n_{\text{D}} : 1$ until it reaches the fully denoised state ($i = 1$). As the model iterates, the state $Y^{(i)}$ is gradually denoised, the fidelity of the approximation of the cost function is improved, and the perturbations to the gradient approximation are decreased such that the model stays in the region of the newfound extrema. Y can be used to represent any optimization variable, in our case, it is directly mapped to the control input.

In the context of conventional diffusion models, this denoising step is performed by solving a stochastic differential equation (SDE). SDE methods have the benefit of maintaining a larger breadth of samples, however, this requires a large and diverse training dataset. Denoising Diffusion Implicit Models (DDIM) [3] and Denoising Diffusion Probabilistic Models (DDPM) [18] are related to the SDE method. DDIM and DDPM provide alternate means of denoising a diffusion model, and have been heavily used in other fields utilizing generative techniques.

III. METHOD

A. Model-Based Diffusion

In contrast to the aforementioned methods, Model-Based Diffusion instead uses a Monte Carlo Score Ascent (MCSA) to guide the denoising process. MCSA is simply stochastic gradient ascent on reward likelihood, as described in equations 6-8. This provides sampling only in the high-likelihood region, as well as sampling at runtime, reducing the number of samples required and eliminating the need for a training dataset [10].

While MCSA can be used in a standalone fashion as an approximation of black-box gradient ascent optimization, it faces the same issues as gradient ascent with regards to local extrema. The diffusion model helps overcome this issue. In the case of MBD, the score distribution, p_i, is gradually corrupted as i is increased. For a nonconvex target distribution p_0, this means that for high i, the distribution will be made increasingly convex, allowing the Monte Carlo score ascent to reach the region of the global optimum prior to the inclusion of local nonconvex features. This is illustrated Fig. 2.

In MBD, the target distribution p_0 is defined as

$$p_0(Y^0) \propto e^{-\frac{J(Y^0)}{\lambda}} \tag{3}$$

where λ is known as the sampling temperature. λ is a hyperparameter which can be tuned to adjust the sampling behavior of the planner. This relation to the cost function $J(Y)$ allows us to link the target distribution back to the cost of the trajectory optimization problem.

The standard MBD algorithm is shown below, from Pan and Yi [10]. Equation (4) describes the constraints on the series of α parameters used in the MBD process. These are solved ahead of runtime, and typically fixed for a given environment. They are linked to the values of σ, which represents the sampling covariance at each timestep.

$$\bar{\alpha}_i = \left(\prod_{k=1}^{i} \alpha_k \right) \forall i \in [1..n_{\text{D}}] \tag{4}$$

$$\sigma^i = \sqrt{\frac{1 - \bar{\alpha}_i}{\bar{\alpha}_i}} \forall i \in [1..n_{\text{D}}] \tag{5}$$

These equations allow the user to work in either direction, either starting with fixed α values — often a linear interpolation with set endpoints [10] — or starting with σ endpoints $\sigma^{n_{\text{D}}}$ and σ^1, obeying the aforementioned constraints on α.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

After calculating the initial parameters, the actual optimization process begins by using (6) in conjunction with (3) to calculate the new weighted mean $\bar{Y}^{(0)}$ of the samples $\mathcal{Y}^{(i)}$. This is equivalent to a weighted average with softmax weights.

$$\bar{Y}^{(0)}(\mathcal{Y}^{(i)}) = \frac{\sum_{Y^{(0)} \in \mathcal{Y}^{(i)}} Y^{(0)} p_0(Y^{(0)})}{\sum_{Y^{(0)} \in \mathcal{Y}^{(i)}} p_0(Y^{(0)})} \tag{6}$$

Equation (7) and (8) then perform score ascent steps, utilizing the previously calculated α values.

$$\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) \approx -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i} \bar{Y}^{(0)}(\mathcal{Y}^{(i)}) \tag{7}$$

$$Y^{(i-1)} = \frac{1}{\sqrt{\alpha_i}} \left(Y^{(i)} + (1-\bar{\alpha}_i) \nabla_{Y^{(i)}} \log p_i(Y^{(i)}) \right) \tag{8}$$

Note that, in practicality, many of these terms cancel, however, they are left in for completeness as (7) is an approximation of the score ascent and not a true equality. The complete algorithm is shown below:

Algorithm 1 Model-based Diffusion

1: **Input:** $Y^{(n_\mathrm{D})} \sim \mathcal{N}(0, I)$
2: **for** $i = n_\mathrm{D}$ to 1 **do**
3: Sample $\mathcal{Y}^{(i)} \sim \mathcal{N}\left(\frac{Y^{(i)}}{\sqrt{\bar{\alpha}_{i-1}}}, (\sigma^{(i-1)})^2) I\right)$
4: Calculate $\bar{Y}^{(0)}$ from (6)
5: Estimate the score $\nabla_{Y^{(i)}} \log p_i(Y^{(i)})$ (7))
6: Monte Carlo score ascent $Y^{(i-1)}$ from (8)
7: **end for**

For a qualitative understanding, the problem can be viewed simply as gradient descent on a function to which detail is iteratively added: One starts with a convex, smooth surface, and takes a step towards the optima. After this step, the algorithm samples a smaller σ^i (5), thus gaining more local information. If the series of σ values is chosen correctly, the local optima will be within the sampled region, allowing the gradient descent to always occur on a locally convex surface.

IV. Importance Sampling

Since the score gradient calculations are a sampling approximation, we can infer that some elements of the gradient are of lower importance. One could form a new orthonormal basis, as demonstrated in [19]. Alternatively, we can use adaptive importance sampling (AIS) to learn the distribution of data which is most likely to improve the score. In this work, we replace the Gaussian sampler of MBD with an adaptive importance sampler.

A. Implementation of Model Based Diffusion with Adaptive Importance Sampling

Typical MBD, for each diffusion step i, specifies an isotropic distribution calculated from σ^i prior to runtime. AIS is an iterative process, which learns the multivariate distribution $d^{(i)}$ by taking a small number of samples per iteration ($n_{s/\mathrm{iter}}$) and improving the estimated distribution.

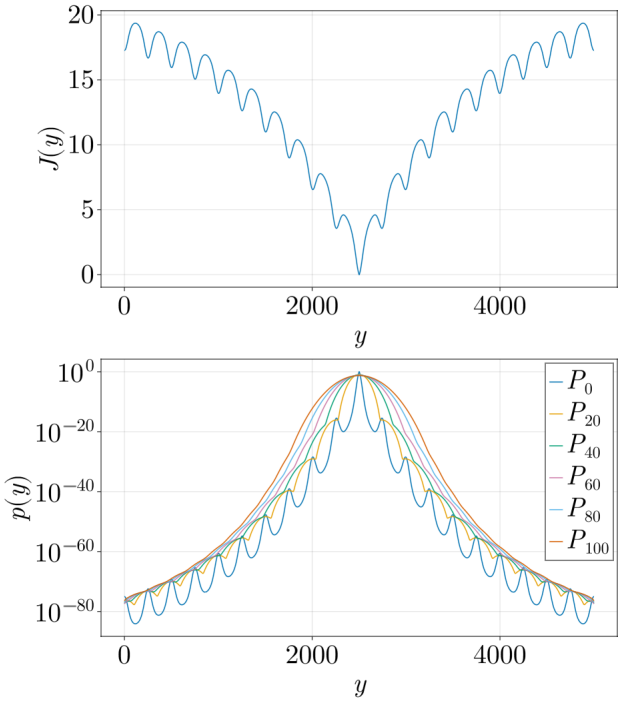

Fig. 2: An image of a synthetic Ackley cost function and it's progressively corrupted score density functions. p_0 is calculated from (3) with successive p_i being plotted as repeated convolution with a Gaussian corresponding to the sample covariances σ. Note that this calculation is not explicitly performed during optimization and is only used for explanatory purposes.

In this implementation, we use the cross-entropy method [20, 21] as our adaptive importance sampler. The cross-entropy algorithm is shown in Algorithm 3. CE was selected as our importance sampler based on results in MPOPI-CE, which showed high performance for small numbers of samples in multi-agent results [11]. Given the similarities between MPOPI and MBD-AIS, we believe that MBD-based results would be similar.

We use the distribution output by the adaptive importance sampler to inform the MCSA step rather than the normal distribution used by [10]. This is shown in Algorithm 2. Note that, while we use the cross-entropy method in this work, there are no limitations on the type of AIS algorithm used, thus we leave line 3 generic.

In this section, we use $d^{(i)}_{(j)}$ to represent the distribution used for cross-entropy adaptation at diffusion step i and CE iteration j. Note that, when this is set equal to another variable, it indicates equality of the distribution itself, not a sampling. For each diffusion step i, we set the initial cross-entropy distribution $d^{(i)}_{(0)} = \mathcal{N}(Y_i, I\sigma^i)$. If we used the result of the previous $(i+1)$ step's adaptation — $d^{(i)}_{(0)} = d^{(i+1)}_{(N_\mathrm{iter})}$ — the problem reduces to cross-entropy with no impact of MBD. This results in worse overall performance and a tendency become stuck in a local minima. By forcibly updating the mean and covariance

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

in accordance with the MBD algorithm, we retain the global optimizing properties of MBD. Interestingly, setting $n_{\mathrm{D}} = 1$ has the same effect, also reducing MBD to the cross entropy method [10]. The use of pure cross-entropy optimization for planning has also previously been demonstrated [22].

Algorithm 2 Model Based Diffusion with Adaptive Importance Sampling

Input: $Y^{(N)} \sim \mathcal{N}(0, I)$
for $i = N$ to 1 **do**
  $d^{(i)}_{(N_{\text{iter}})} \leftarrow \text{AIS}\left(\mathcal{N}\left(\frac{Y^{(i)}}{\sqrt{\bar{\alpha}_{i-1}}}, \left(\frac{1}{\bar{\alpha}_{i-1}} - 1\right) I\right)\right)$
  Sample $\mathcal{Y}^{(i)} \sim d^{(i)}_{(N_{\text{iter}})}$
  Calculate $\bar{Y}^{(0)}$ from (6)
  Estimate the score $\nabla_{Y^{(i)}} \log p_i(Y^{(i)})$ (7))
  Monte Carlo score ascent $Y^{(i-1)}$ from (8)
end for

Algorithm 3 Cross-Entropy Method

Require: Loss function $J(y)$, max iterations N_{iter}, sample size $n_{s/\text{iter}}$, elite threshold τ, minimum elite samples $n_{\min}$
$d_{(0)}$ given
for $j = 1, 2, \ldots, N_{\text{iter}}$ **do**
  $\{s_i\}_{i=1}^{n_{s/\text{iter}}} \sim d_{(j-1)}$
  $\{L_i\}_{i=1}^{n_{s/\text{iter}}} \leftarrow \{J(s_i)\}_{i=1}^{n_{s/\text{iter}}}$
  Sort samples so that $L_{(1)} \leq L_{(2)} \leq \cdots \leq L_{(n_{s/\text{iter}})}$
  if $L_{(n_{s/\text{iter}})} < \tau$ **then**
    $n_{\text{elite}} \leftarrow n_{s/\text{iter}}$
  else
    $n_{\text{elite}} \leftarrow \max\Big(n_{\min}, \min\{\, i : L_{(i)} > \tau\} - 1\Big)$
  end if
  $\mathcal{E} \leftarrow \{\, s_{(1)}, \ldots, s_{(n_{\text{elite}})}\}$
  $d_{(j)} \leftarrow \text{fit}(d_{(j-1)}, \mathcal{E})$ ▷ Use Maximum Likelihood Estimation to fit distribution to elites
end for
return $d_{(N_{\text{iter}})}$

Note that MPOPI-CE uses the method proposed by Schäfer and Strimmer [23] to fit distributions, consistent with the original MPOPI-CE repository, while our method uses Maximum Likelihood Estimation.

B. Bad samples are worse than no samples

From the above algorithm, one might intuit that it would be best to reuse all samples used in the sampler adaptation in addition to those drawn from the final distribution:

$$\mathcal{Y}^{(i)} = \left\{s_1, s_2, ..., s_{n_{s/\text{iter}}}, y\right\}, y \sim d^{(i)}. \tag{9}$$

Implementation of this proved unfruitful, with this algorithm providing suboptimal results, nearly never converging. We believe this to be a product of the unusual sample distributions found during the adaptation – since the sample distribution is learned, it can often sample regions far from the optimum, resulting in worse performance when used in diffusion.

V. Experimental Simulation Design

A. Overview

In this section we use numerical simulations to evaluate MBD and MBD-AIS. In addition, we evaluate the performance of similar algorithms such as Model Predictive Path Integral Control (MPPI) and Model Predictive Optimized Path Integral Control-Cross Entropy (MPOPI-CE).

B. Setup

For all cases, MPPI [12] and MPOPI-CE are implemented from the repository given by Asmar [11] and use identical parameters, namely, $\lambda = 10$, CE iterations $N_{\text{iter}} = 10$, and elite samples $n_{min} = 0.2 n_{s/\text{iter}}$ (rounded to the nearest integer). MBD and MBD-AIS are implemented according to algorithms 2 and 3. For MPPI and MPOPI-CE, the effective number of samples are simply the total number of samples used by one iteration of the MPPI/MPOPI-CE algorithm. For MBD, we divide the number of effective samples by the number of diffusion iterations.

$$n_{\text{samp}} = \frac{n_{\text{eff}}}{n_{\mathrm{D}}} \tag{10}$$

This maintains a fixed number of model evaluations between both algorithms, ensuring fair comparisons on the basis of required data rather than runtime. Our parameters for the MBD-AIS CE sampler are set to a fixed $N_{\text{iter}} = 2$ and a minimum number of elite samples $n_{min} = 2$. The elite threshold is set to the value of the reward expected for the relevant environment. The number of samples per iteration are chosen as follows, where $\lfloor \cdot \rfloor$ is the floor operator.

$$n_{s/\text{iter}} = \left\lfloor \frac{n_{\text{samp}}}{N_{\text{iter}} + 1} \right\rfloor \tag{11}$$

Once the MBD-AIS CE algorithm completes its iterations, the final number of samples remaining to be passed to the diffusion algorithm is:

$$n_{\text{final}} = n_{\text{samp}} - n_{s/\text{iter}} N_{\text{iter}}. \tag{12}$$

This step is designed such that all parts of the process have an approximately equal number of samples, with the balance going to diffusion. Note that for large n_{D}, $n_{s/\text{iter}}$ will fall below the minimum number of elite samples, meaning that MBD-AIS cannot be used with this configuration. Unless otherwise specified, we use the following hyperparameters:

$$\begin{cases} n_{\mathrm{D}} & = 100 \\ \sigma^{n_{\mathrm{D}}} & = 0.6 \\ \sigma^{1} & = 10^{-2} \\ \lambda & = 0.1. \end{cases} \tag{13}$$

We found empirically that MBD-AIS was relatively insensitive to parameters used. As is demonstrated in VI, we tested a wide range of n_{D}. We also varied λ, both values of σ, and the cross-entropy parameters $n_{s/\text{iter}}$, N_{iter}, and n_{min} prior to deciding on experimental design. We chose $n_{min} = 2$ since this is the minimum required to fit a distribution in CE. Setting

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

this value as small as possible allows us to use the widest range of n_{eff} without adjusting sampler parameters, and we similarly choose $N_{\text{iter}} = 2$ in order to maximize valid range. Further work should investigate tuning of these parameters in configurations for high n_{eff}, as budgets for N_{iter} and $n_{s/\text{iter}}$ can be drastically changed when not operating in a small sample range.

C. Environments

The first experiment conducted is the car racing environment provided by Asmar [11]. This provides an optionally multi-agent environment with cars attempting to maximize speed around a track. There is sufficient curvature to necessitate a long horizon. 50 timesteps are used in the original paper in a receding horizon scheme. In order to demonstrate the long-horizon performance of MBD, we use 200 timesteps as described below. The car racing environment is evaluated 25 times per permutation of sample count and policy type. The reward for this environment is structured as

$$R(\mathbf{x}_t) = 2|v_t| - |d(\mathbf{x}_t)| - 5000R_\beta(\mathbf{x}_t) - 10^6 R_t(\mathbf{x}_t), \quad (14)$$

wherein $\mathbf{x}_t$ indicates state, v represents velocity, d represents distance from the track centerline, and R_β and R_t are boolean values indicating violations of sideslip angle and track boundaries, respectively. For more detail on this environment, see [11]. For the car racing experiment, we compare MBD and MBD-AIS with MPPI and MPOPI-CE. An image of the track with representative trajectories is shown in Fig. 1.

Additionally, we implement the Ant-V4 and HalfCheetah-V4 cases from MuJoCo [24], contrasting the results of MBD-AIS and MBD. These are each run for 10 simulations. Note that we run a single, long-horizon plan, and do not perform a receding horizon scheme. For the Ant-V4 and HalfCheetah-V4 experiments, we compare MBD with MBD-AIS. Given the poor results of CE-MPPI and MPPI in long-horizon planning, and prior comparisons of MBD and MPPI in these cases [10], we did not repeat these experiments with MPPI.

VI. Results

A. Car Racing Environment

The MBD-AIS CE sampler drastically improves performance when compared to the standard sampler. This is visible in Fig. 4 and Fig. 3, as well as their corresponding tables. Throughout tables and figures, standard MBD is referred to as only MBD, with our version being referred referred to as MBD-AIS. For both, they are followed by their corresponding n_{D}.

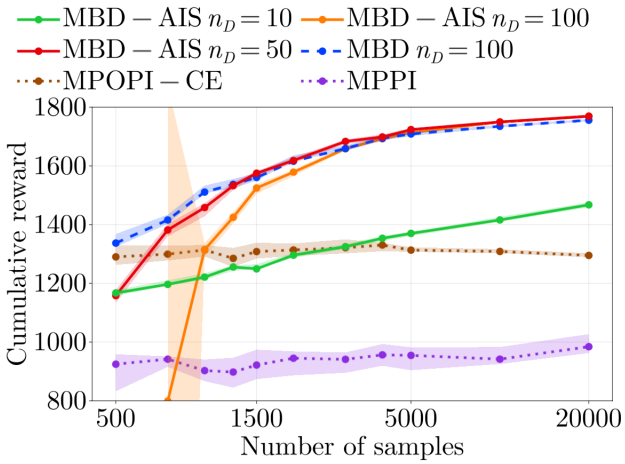

Fig. 3: Rewards for the 1 car racing task from Asmar [11] with a fixed horizon of 50 timesteps.

n_{samp}	Rewards MPPI	MPOPI-CE	MBD $n_D = 100$	MBD-AIS $n_D = 100$
500	925±126	1290±66	**1337±44**	
750	942±36	1299±51	**1416±32**	799±4189
1000	903±74	1314±39	**1511±27**	1315±48
1250	898±102	1285±65	**1537±31**	1425±38
1500	922±100	1308±54	**1561±26**	1525±33
2000	945±82	1314±42	**1616±19**	1579±21
3000	941±69	1321±48	**1659±18**	1658±13
4000	956±74	1330±29	**1694±14**	1693±14
5000	954±78	1313±16	1709±13	**1713±9**
10000	942±58	1309±14	1735±9	**1750±8**
20000	984±64	1295±13	1756±8	**1769±5**

TABLE I: 1 car 50-step horizon MBD results.

In this section, we define sampling efficiency by picking a given reward for the baseline algorithm (in this case, MBD) and calculating the interpolated number of samples required for the same reward in our algorithm. We call the interpolator $g(R)$, where R is the reward. Thus, we define sample efficiency as

$$\eta(R) = \frac{g_{\text{MBD}}(R)}{g_{\text{MBD-AIS}}(R)}. \quad (15)$$

For example, in Table I, MBD requires 20000 samples for a reward of 1756. MBD-AIS requires $g_{\text{MBD-AIS}}(1756) = 10000 + \frac{20000-10000}{1769-1750}(1756 - 1750) = 13158$ samples for the given reward, yielding $\eta_{samp}(1755) = \frac{20000}{13158} = 1.52$.

Note that in Fig. 3, MPPI outperforms MBD for a small number of samples and that MBD-AIS and MBD perform very similarly. If the degrees of freedom are increased (2 cars rather than 1), as in Fig. 4, the benefit of MBD-AIS becomes more apparent. Shown in Tables I and II are the data for the corresponding figures. Intervals shown are the 95% confidence interval, as calculated from the sample size of 25 runs per case. For the 2-car case, we find a maximum sample efficiency improvement of $\eta = 11.2$.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

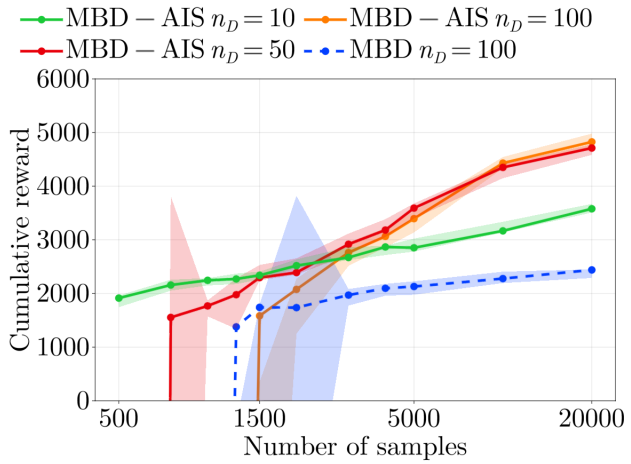

Fig. 4: Rewards for the 2 car racing task from Asmar [11] with a fixed horizon of 200 timesteps. MPPI and MPOPI-CE have rewards below zero (indicating a constraint violation) and are thus removed from the plot. Notably, MBD-AIS outperforms standard MBD for all cases.

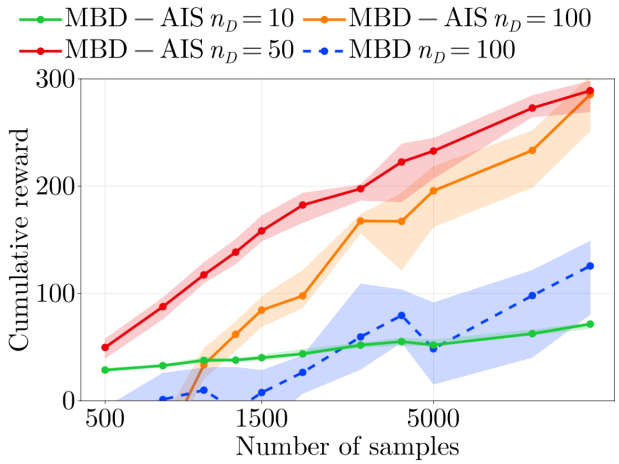

Fig. 5: Plot of MBD reward in the MuJoCo HalfCheetah-V4 case with respect to number of samples, showing high MBD-AIS performance.

n_{samp}	Rewards			
	MBD $n_D = 100$	MBD-AIS $n_D = 10$	MBD-AIS $n_D = 50$	MBD-AIS $n_D = 100$
500	−166759	**1914±218**	−150752	
750	−101083	**2158±203**	1553±22956	−482478
1000	−25540	**2245±137**	1768±297	−190491
1250	1378±9057	**2271±195**	1977±941	−20134
1500	1741±9832	**2339±146**	2295±285	1583±9345
2000	1736±7883	**2519±161**	2391±305	2076±1023
3000	1970±309	2671±242	**2918±393**	2763±435
4000	2097±221	2867±176	**3182±332**	3064±356
5000	2130±250	2852±241	**3592±278**	3395±393
10000	2277±211	3168±198	4350±263	**4431±210**
20000	2439±162	3579±162	4712±195	**4826±250**

TABLE II: 2 car 200-step horizon MBD results. MPPI neglected due to nonconvergence. Negative values (marked in red) indicate failure to meet constraints and have large confidence intervals, thus their CI terms are neglected.

B. MuJoCo Environments

In addition to the car racing environment, we use MuJoCo test cases including Ant-V4 and HalfCheetah-V4 implemented via EnvPool and PyCall in Julia. As expected, the results show similarly favorable trends for the MBD-AIS cases, nearly always outperforming base MBD in the cases for which the n_{eff}:n_{D} ratio is sufficiently high.

C. HalfCheetah-V4

The HalfCheetah-V4 environment is implemented as provided by MuJoCo [24]. As shown in Fig. 5, MBD-AIS outperforms standard MBD in all but the $n_{\text{D}} = 10$ case. MBD-AIS reaches a higher reward across the range of samples. We find a maximum sample efficiency improvement of $\eta = 13.6$. The corresponding numeric results are shown in Table III.

n_{samp}	Rewards			
	MBD $n_D = 100$	MBD-AIS $n_D = 10$	MBD-AIS $n_D = 50$	MBD-AIS $n_D = 100$
500	−18±37	29±3	**50±19**	
750	1±47	33±3	**88±20**	−44±37
1000	10±52	38±5	**117±20**	34±29
1250	−9±56	38±3	**139±24**	62±27
1500	8±54	40±6	**158±24**	85±29
2000	26±36	44±6	**182±28**	98±35
3000	60±80	52±5	**198±15**	168±18
4000	80±50	55±7	**223±55**	167±73
5000	48±77	52±6	**233±38**	196±57
10000	98±82	63±6	**273±21**	233±54
15000	126±69	71±6	**289±29**	285±52

TABLE III: HalfCheetah-V4 rewards for MBD and MBD-AIS.

D. Ant-V4

The Ant-V4 environment is implemented as provided by MuJoCo [24]. We find that MBD-AIS again outperforms MBD. Even in the lowest-sample configuration, MBD-AIS reaches higher rewards than the highest tested n_{samp} for standard MBD. This is shown in Fig. 6 and Table IV. Because the resulting rewards do not overlap, we do not define a sample efficiency for this experiment, noting instead that we achieve significantly higher rewards.

n_{samp}	Rewards			
	MBD $n_D = 100$	MBD-AIS $n_D = 10$	MBD-AIS $n_D = 50$	MBD-AIS $n_D = 100$
500	−101	**273±25**	152±44	
750	−75	**284±23**	164±53	53±63
1000	−90	**282±10**	187±45	68±62
1250	−92	**280±24**	166±47	59±60
1500	−105	**286±17**	166±37	67±40
2000	−65	**290±20**	179±44	68±49
3000	−94	**289±20**	191±55	64±52
4000	−62	**299±16**	206±52	77±58
5000	−44±53	**308±27**	207±52	104±67
10000	−92	**301±8**	206±61	97±53
15000	−49±90	**320±19**	217±47	68±90

TABLE IV: Ant-V4 rewards for MBD and MBD-AIS.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

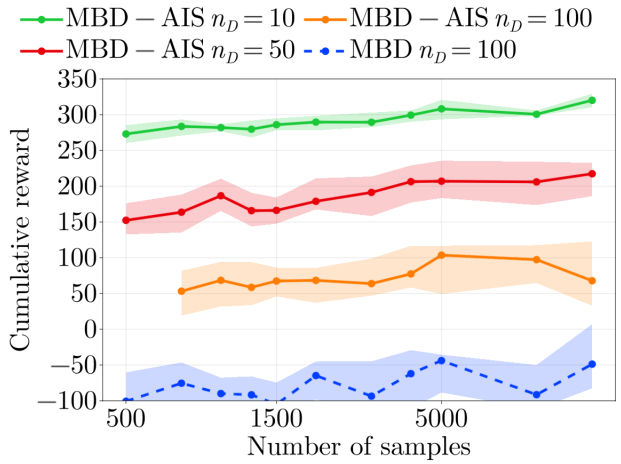

Fig. 6: Plot of MBD reward in the MuJoCo Ant-v4 case with respect to number of samples, showing high MBD-AIS performance.

VII. Discussion

In all three of the higher degree-of-freedom experiments, MBD-AIS outperforms basic MBD. We demonstrate as much as a 13x improvement in sample efficiency. We also demonstrate feasible, though suboptimal, planning for cases such as Fig. 4 where standard MBD does not find a feasible (positive reward) path.

Notably, our most complex task, Ant-V4 , with an input space of $Y^{(0)} \in \mathcal{R}^{2000}$ ($u_t \in \mathcal{R}^8$, $T = 250$), shows the starkest difference between MBD-AIS and standard MBD. For the range of samples tested, MBD does not even approach the values of the cross-entropy sampler.

We find that for comparatively low degree-of-freedom problems and a low quantity of samples, such as the one-car short-horizon problem in Fig. 3, MPOPI-CE outperforms MBD and cross-entropy MBD in terms of sample efficiency. We believe this to be due to the simplicity of the problem: MPPI, MPOPI-CE, and MBD are strong algorithms for such simple tasks, which have easily-reachable maximum rewards. Similarly, for very low sample count, MBD-AIS does not have sufficient samples to learn the reward distribution. However, as both the number of samples and degrees of freedom increase, we find that MBD-AIS outperforms MPOPI-CE, MPPI, and MBD, demonstrating both a faster rise time and higher final asymptote.

We also find a consistent trend in performance of the varying n_D configurations. For the HalfCheetah-V4 and car experiments, increasing the number of diffusion steps for a large n_eff results in increased reward. Given the rising slopes of the $n_\mathrm{D} = 100$ and $n_\mathrm{D} = 50$ configurations of the Ant-V4 experiment, we believe that this pattern would hold if this experiment was run with more samples. Conversely, we show that for a small number of samples, lowering the number of diffusion steps proves more effective. We believe that this is due to the number of samples required for the adaptive importance sampler to effectively fit its intermediate distributions. Standard MBD or even MPPI/MPOPI-CE may also provide better results in very simple problems or low sample counts due to this limitation.

As our results show, we expect MBD-AIS to outperform MBD when there is sufficiently high n_samp, which can be varied by adjusting the combination of total sample budget n_eff and diffusion steps n_D. Further experimentation on a more diverse set of experiments and range of time horizons would be required to construct a generalized empirical recommendation of which configuration to use. We posit that it will be heavily influenced by the size of $Y^{(0)}$, convexity, and stiffness of the environment.

While our method significantly improves sample efficiency, it should be noted that there is an effect on the parallelizability of the code. Related methods, such as MPPI, are embarrassingly paralallelizable, with 100% of samples able to be rolled out at the same time. MBD is able to rollout a maximum of n_samp samples at a time, while MBD-AIS is only able to roll out $n_{s/\mathrm{iter}}$. This may affect realtime performance, particularly for hardware-accelerated systems.

VIII. Conclusion

A. Conclusion

In this work, we demonstrate effective usage of adaptive importance sampling in conjunction with Model-Based Diffusion utilizing cross-entropy as our adaptive importance sampler. For cases with a long time horizon and a sufficiently large action space, MBD-AIS outperforms MBD with as few as $\frac{1}{13}$ as many samples. For cases with a very small number of available samples, we show that it is more effective to reduce the number of diffusion steps and increase the number of samples per iteration. The above trends hold for our tests in the car racing environment as well as MuJoCo multijoint physics environments.

IX. Acknowledgment

*This work was supported by Sandia National Laboratories and the United States Air Force Research Laboratory (AFRL). Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC (NTESS), a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration (DOE/NNSA) under contract DE-NA0003525. This written work is authored by an employee of NTESS. The employee, not NTESS, owns the right, title and interest in and to the written work and is responsible for its contents. Any subjective views or opinions that might be expressed in the written work do not necessarily represent the views of the U.S. Government.

*This work is sponsored by the Air Force Research Laboratory, Munitions Directorate (RWTA), Eglin AFB, FL under contract A2308-057-089-003242. Opinions, findings and conclusions, or recommendations are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

REFERENCES

[1] C.R. Hargraves and S.W. Paris. "Direct trajectory optimization using nonlinear programming and collocation". In: *Journal of Guidance, Control, and Dynamics* 10.4 (1987), pp. 338–342. DOI: 10.2514/3.20223.

[2] John T. Betts. "Survey of Numerical Methods for Trajectory Optimization". In: *Journal of Guidance, Control, and Dynamics* 21.2 (1998), pp. 193–207. DOI: 10.2514/2.4231.

[3] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[4] Jacob Austin et al. "Structured denoising diffusion models in discrete state-spaces". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 17981–17993.

[5] Jonathan Ho et al. "Imagen video: High definition video generation with diffusion models". In: *arXiv preprint arXiv:2210.02303* (2022).

[6] Chiyu Jiang et al. "Motiondiffuser: Controllable multi-agent motion prediction using diffusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 9644–9653.

[7] Taeyoung Yun et al. "Guided Trajectory Generation with Diffusion Models for Offline Model-based Optimization". In: *arXiv preprint arXiv:2407.01624* (2024).

[8] Anjian Li et al. "Efficient and Guaranteed-Safe Non-Convex Trajectory Optimization with Constrained Diffusion Model". In: *arXiv preprint arXiv:2403.05571* (2024).

[9] Joao Carvalho et al. "Motion planning diffusion: Learning and planning of robot motions with diffusion models". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 1916–1923.

[10] Chaoyi Pan et al. *Model-Based Diffusion for Trajectory Optimization*. 2024. arXiv: 2407.01573 [cs.RO]. URL: https://arxiv.org/abs/2407.01573.

[11] Dylan M Asmar et al. "Model predictive optimized path integral strategies". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 3182–3188.

[12] Grady Williams et al. "Aggressive driving with model predictive path integral control". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1433–1440. DOI: 10.1109/ICRA.2016.7487277.

[13] Steven LaValle. "Rapidly-exploring random trees: A new tool for path planning". In: *Research Report 9811* (1998).

[14] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.

[15] Yuval Tassa, Tom Erez, and William Smart. "Receding horizon differential dynamic programming". In: *Advances in neural information processing systems* 20 (2007).

[16] Takayuki Osa et al. "An algorithmic perspective on imitation learning". In: *Foundations and Trends® in Robotics* 7.1-2 (2018), pp. 1–179.

[17] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[19] E. Patelli and H. J. Pradlwarter. "Monte Carlo gradient estimation in high dimensions". In: *International Journal for Numerical Methods in Engineering* 81.2 (2010), pp. 172–188. DOI: https://doi.org/10.1002/nme.2687.

[20] Zdravko I Botev et al. "The cross-entropy method for optimization". In: *Handbook of statistics*. Vol. 31. Elsevier, 2013, pp. 35–59.

[21] Reuven Y Rubinstein. "Optimization of computer simulation models with rare events". In: *European Journal of Operational Research* 99.1 (1997), pp. 89–112.

[22] Marin Kobilarov. "Cross-entropy motion planning". In: *The International Journal of Robotics Research* 31.7 (2012), pp. 855–871. DOI: 10.1177/0278364912444543.

[23] Juliane Schäfer and Korbinian Strimmer. "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics". In: *Statistical applications in genetics and molecular biology* 4.1 (2005).

[24] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.