

Low Frequency Sampling in Model Predictive Path Integral Control

Bogdan Vlahov¹, Jason Gibson¹, David D. Fan², Patrick Spieler²,
Ali-akbar Agha-mohammadi², Evangelos A. Theodorou¹

Abstract—Sampling-based model-predictive controllers have become a powerful optimization tool for planning and control problems in various challenging environments. In this paper, we show how the default choice of uncorrelated Gaussian distributions can be improved upon with the use of a colored noise distribution. Our choice of distribution allows for the emphasis on low frequency control signals, which can result in smoother and more exploratory samples. We use this frequency-based sampling distribution with Model Predictive Path Integral (MPPI) in both hardware and simulation experiments to show better or equal performance on systems with various speeds of input response.

Index Terms—Optimization and Optimal Control; Motion and Path Planning; Integrated Planning and Control

I. INTRODUCTION

AS autonomous systems grow in interest, the choice of methods and algorithms used to do real-time motion planning and control becomes critical to achieve complex tasks. In general, there are two approaches to this sort of problem: gradient-based and sampling-based. Gradient-based approaches, such as Differential Dynamic Programming (DDP) [1] or Sequential Quadratic Programming (SQP) [2], generally have requirements on the dynamics or cost functions used such as they need to be continuously differentiable. In exchange, they provide controls that converge on the true optimal sequence as the number of iterations increase. Sampling-based methods such as [3], [4] relax the requirements on the dynamics and cost functions to allow for completely arbitrary functions but require many samples to get a good estimation of the true optimal control trajectory.

Model Predictive Path Integral (MPPI) is one such sampling-based algorithm that has been used to achieve aggressive behavior in a small-vehicle setting. While ideally, one would sample every possible control trajectory to determine the true path integral, this suffers from the curse of dimensionality as both the control dimensions and the time horizon



Fig. 1. (Upper) The off-road vehicle in a desert terrain just before an autonomy test. (Lower) A screenshot of the Flightmare quadrotor simulator

increase. By instead sampling from a Gaussian distribution, computationally-feasible solutions has been derived in path-integral [4], information-theoretic [5], and stochastic search [6] approaches. The Gaussian distribution gives a clean equation for distribution calculations and is a natural starting point when constructing a sampling-based algorithm.

However, sampling a Gaussian at every time step leads to control trajectories samples containing high-frequency noise. Depending on the dynamics model, the effect of the high frequency controls can be dampened when calculating the state trajectory, allowing for control trajectories with high-frequency noise and low-frequency noise to produce similar costs. This in turn allows the approximate optimal control trajectories computed from finite samples to chatter significantly. This can cause damage over time when applied to real systems. When looking at data collected from human experts on these systems, it can be seen that they generally take smoother actions over a longer time horizon to achieve their desired behavior.

In order to address these issues, we look to sampling distributions that can adjust the level of high-frequency noise that appears in control trajectories. In this paper, we explore sampling focused around the low-frequency domain by using a colored noise distribution. We show how this choice of sampling distribution can lead to better state exploration when

Manuscript received: October, 12, 2023; Accepted March, 11, 2024.

This work was supported in part by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (80NM0018D0004) and the Defense Advanced Research Projects Agency, and in part by the Office of Naval Research (N00014-21-1-2074). (Corresponding author: Bogdan Vlahov)

¹ Bogdan Vlahov, Jason Gibson, and Evangelos Theodorou are with the Autonomous Control and Decision Systems Lab, Georgia Institute of Technology, Atlanta GA 30313 USA (e-mail: bvlahov3@gatech.edu; jgibson37@gatech.edu; evangelos.theodorou@gatech.edu)

² David D. Fan, Patrick Spieler, and Ali-akbar Agha-mohammadi are with NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA (e-mail: david.fan@gmail.com; patrick.spieler@jpl.nasa.gov; aliagha4@gmail.com)

compared to standard Gaussian sampling in MPPI, like shown in Fig. 2. Furthermore, we will show the resulting controls are smoother and thus reduce the wear and tear on systems, such as those shown in Fig. 1, caused by chattering. Previous works have attempted to sample smoother controls in various ways but our method provides an additional parameter, γ , to adjust the smoothness of sampled control without adjusting the overall range of sampled controls like adjusting variance would cause. This smoothness adjustment is done by explicitly lowering the chance of sampling of higher frequency signals, resulting in less oscillatory behavior and allows its use with multiple systems with different control bandwidths. The contributions of this paper can be summarized as follows:

- We show how a frequency-based sampling distribution can be used in MPPI with minimal adjustments to the update rules and optimal control calculation in Section III.
- We perform experiments on a real hardware platform that has large control lag in Section IV-A as well as highly-reactive systems in Sections IV-B and IV-C. These tests show that our approach can be considered a generalization of Gaussian sampling applicable in many scenarios.

The paper is organized as follows. In Section II, we go over other approaches to choosing sampling distributions. In Section III, we discuss the frequency-based sampling is performed and the modifications required to MPPI. We showcase experiments in Section IV and conclude in Section V.

II. RELATED WORK

Before going into frequency-based techniques, a brief overview of relevant concepts is required. The Power Spectral Density (PSD) provides a measure of how much power every frequency, f , contributes to a time-domain signal. Samples from a zero-mean, uncorrelated Gaussian distribution have constant power at every frequency, $PSD(f) \propto 1$. Colored Noise distributions cover a large space of distributions that have a PSD of the form $PSD(f) \propto \frac{1}{f^\gamma}$.

There have been multiple works on adjusting the sampling distributions used in MPPI to improve sampling exploration. The original Gaussian sampling distribution used in [4] provides a good starting point, but can have difficulty to fully explore the possible state space. [7] uses Gaussian Processes (GPs) as their sampling distribution and can show that the resulting optimal controls are smoother than those found using Gaussian noise. However, it could require much more computation to generate samples depending on the time horizons chosen as the prior. [8] uses a Normal Log-Normal (NLN) distribution for sampling and shows that the samples generated can explore a wider space than standard Gaussian sampling. While NLN distributions allow for more sampling on the tail ends of a distribution compared to a normal distribution, there is still only a small chance to sample multiple tail-end values in a row for slow-acting systems unlike colored noise distributions. [9] showed that sampling in a derivative "action space" would also further improve the exploration and smoothness of the control trajectories MPPI would produce. Specifically, the authors end up not integrating the Gaussian samples over time but instead over iterations of MPPI. While

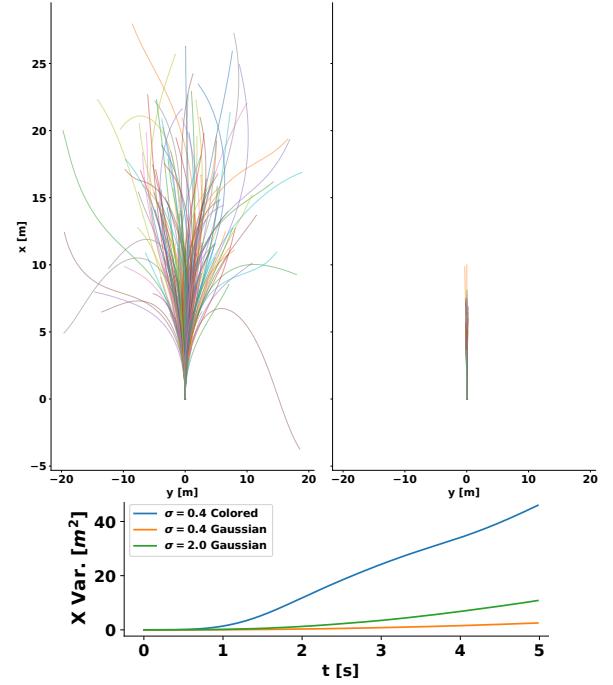


Fig. 2. 300 samples of state trajectories generated from running frequency-based (Left) and Gaussian (Right) control samples through the off-road vehicle dynamics described in Section IV-A. The frequency-based throttle and steering samples are generated using $\gamma = 4$ and both distributions used the same σ set to 0.4. Descriptions of γ and σ can be found in Section III-A. (Lower) Variance in the x direction of the same 300 samples from frequency-based and Gaussian distributions over time. There is an additional Gaussian distribution with $\sigma = 2.0$ to show that increasing the standard deviation does not allow Gaussian sampling to explore as far as colored samples. Colored samples reach further extremes in both the x and y directions due to the reduction of high frequency signals in the control trajectories.

this sampling technique does improve smoothness, it does so by multiplying the samples by dt . This can cause exploration issues with small dt . Our frequency-based sampling can adjust the time correlation of our trajectories independent of dt .

Other works have used multi-hypothesis distributions [10], [11] such as Gaussian Mixture Models (GMMs) and Stein Variational distributions to try to increase the exploration capabilities of the algorithm. While GMMs can maintain multiple distributions, there is nothing to prevent the several modes from collapsing which became the motivation for Stein variational policies. Stein variational policies maintain multiple particles as Gaussian distributions with fixed variance but ensure that the means of these Gaussians are pushed apart by inclusion of a kernel function. Recently, there has also been work in learning sampling distributions with concepts like normalizing flow [12], [13]. These methods make use of a starting distribution and then perform a transform of that base distribution into a new one. This transform can then make use of machine learning to incorporate outside information such as the location of obstacles in order to bias the resulting distribution into safe regions.

The use of frequency-based sampling has been explored before as well. [14] used frequency-based sampling as well as other techniques in a Cross-Entropy Method (CEM) controller to create improved Cross-Entropy Method (iCEM). They

showed that using frequency-based sampling has one of the largest positive impacts on the overall performance of the CEM controller. We will use the same frequency-based sampling technique with MPPI which uses a weighted combination of every sample to compute the optimal control trajectory rather than using a percentage cutoff. [15] used a power law noise to improve exploration for their Reinforcement Learning (RL) agent. They specifically generated a $\frac{1}{f^2}$ distribution by filtering white noise and saw large improvements over Gaussian sampling in their ablation study. We will be using a different power law noise generation technique so that we can use any exponent we desire.

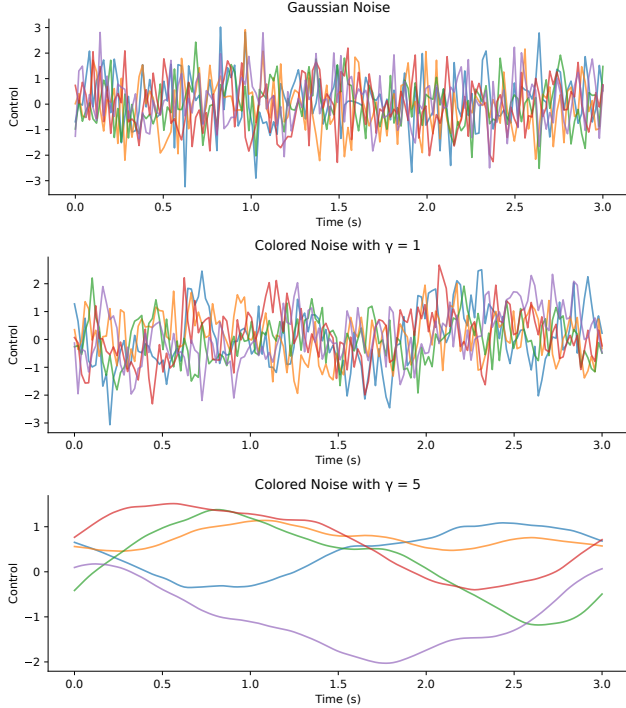


Fig. 3. A comparison of Gaussian and Colored noise with various exponents. As γ increases, the samples generated become smoother and can reach tail-end values more consistently.

III. MATHEMATICAL BACKGROUND

A. Low Frequency Sampling Distribution

For this paper, we will be using a colored noise distribution found by sampling Gaussians in the frequency domain and performing the inverse Fast Fourier Transform (iFFT) to go to the time domain, first described in [16]. These samples are control trajectories with a finite time horizon of T steps. We first will show that samples derived from this method are still Gaussian in the time domain.

In order to limit our time-domain signal to contain only real components, we sample a particular frequency-domain sequence known as a Hermitian-symmetric sequence [17, Fig. 2.5]. A Hermitian-symmetric sequence is one where the second half of the sequence is the complex conjugate of the first half of the sequence. Let $Z[n] = Z_{real}[n] + iZ_{imag}[n]$ be a point in the frequency domain with a real and imaginary component. There will be $N = \lfloor \frac{T}{2} \rfloor + 1$ frequency samples.

The sampling of $Z_{real}[n]$ and $Z_{imag}[n]$ is done from Gaussian distributions of the form $\mathcal{N}(\mu_n, (\max\{\frac{n}{N}, f_{min}\})^{-\gamma} \frac{\sigma^2}{\zeta})$ where $f_{min} = \frac{1}{N}$ is a cutoff frequency to ensure non-zero variance, γ is a user-chosen colored exponent, σ is a user-chosen standard deviation, and

$$\zeta = T^{-2} N^\gamma \left(1 + 4 \sum_{n=1}^{N-1} n^{-\gamma} \right) \quad (1)$$

ensures that the time-domain variance equals σ^2 . The effects of different γ on sample control trajectories is shown in Fig. 3. The Hermitian-symmetric sequence can be compactly represented as $Z_N = \{Z[0], Z[1], \dots, Z[N-1]\}$ in the frequency domain. As each point $Z[n]$ in the frequency domain has both a real and imaginary part, there is no reduction of the total number of points sampled compared to uncorrelated time-domain Gaussian sampling. The full frequency-domain sequence, Z'_T can then be constructed as

$$Z'_T[t] = \begin{cases} Z[t] & \text{if } 0 \leq t \leq N-1 \\ \overline{Z[T-t]} & \text{if } N-1 < t < T \end{cases}, \quad (2)$$

where $\bar{\cdot}$ is the conjugate operator. The time-domain sequence is found using the iFFT, referred to as Ψ for notional convenience:

$$z(t) = \frac{1}{T} \sum_{n=0}^{T-1} Z'_T[n] e^{i2\pi nt/T} = \Psi(Z_N). \quad (3)$$

The iFFT is thus a matrix transform of our frequency-domain samples and can be simplified to the following when sampling Hermitian-symmetric sequences:

$$\begin{bmatrix} z(0) \\ \vdots \\ z(t) \\ \vdots \end{bmatrix} = M \begin{bmatrix} Z_{real}[0] \\ \vdots \\ Z_{real}[n] \\ Z_{imag}[n] \\ \vdots \end{bmatrix}, \quad \forall \begin{matrix} n = 0, \dots, N-1 \\ t = 0, \dots, T-1 \end{matrix} \quad (4)$$

$$M = \frac{1}{T} \begin{bmatrix} 1 & \dots & 2 & 0 & \dots \\ & & \vdots & & \\ 1 & \dots & 2 \cos\left(\frac{2\pi nt}{T}\right) & -2 \sin\left(\frac{2\pi nt}{T}\right) & \dots \\ & & \vdots & & \end{bmatrix}. \quad (5)$$

Note that if T is even, the final sample, $Z[N-1]$, must only have a real component ($Z_{imag}[N-1] = 0$) to be considered Hermitian. As the M matrix does not depend on the samples drawn for $Z[n]$, each $z(t)$ is a linear combination of Gaussian independent random variables, making it Gaussian as well. However, these samples will now be time-correlated. In a slight abuse of notation, we will use the term Gaussian to mean uncorrelated Gaussian distributions and Colored to refer to our frequency-based sampling technique throughout the rest of this paper unless otherwise noted.

B. MPPI derivation from Frequency Domain Sampling

Consider a general nonlinear system with discrete-time dynamics and cost function of the following form:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t) \Delta t + \mathbf{w}_t, \quad (6)$$

$$\mathbf{J}(X, U) = \phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} q(\mathbf{x}_t), \quad (7)$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is our state, $\mathbf{u} \in \mathbb{R}^{n_u}$ is our control, and $\mathbf{w} \in \mathbb{R}^{n_x}$ are external disturbances that are unknown but assumed to be bounded. $\mathbf{J}(\cdot, \cdot)$ is the cost function of a given state trajectory X , and control trajectory U , with a terminal cost $\phi(\cdot)$ and nonlinear state cost $q(\cdot)$. We exclude control costs from this work for the sake of brevity but they can be added with no change to the following discussion.

MPPI is trying to minimize the cost function of the system, by sampling and applying a weighted exponential averaging to produce the optimal control. By following the stochastic search derivation of MPPI, we can make some small adjustments to accommodate our new sampling distribution. We make use of the fact that Gaussian distributions belong to the exponential family of probability distributions P , parameterized by θ , with $\theta = [\mu, \Sigma]^T$ specifically for Gaussians. We define $l(\theta)$ in order to better align with theory presented in [6],

$$l(\theta) = \ln \mathbb{E}_{\epsilon \sim P(\theta)} [\mathbf{S}(-\mathbf{J}(X, U))]. \quad (8)$$

where $\mathbf{S}(\cdot)$ is a generic shaping function. This function $l(\theta)$ represents a log-transform on the expectation of the negated cost function over sampled controls. Maximizing $l(\theta)$ is equivalent to minimizing $\mathbb{E}[\mathbf{J}(X, U)]$ which is the goal of our original optimal control problem. [6] shows that, given $l(\theta)$, the update law for θ at iteration k can be gradient ascent,

$$\theta_t^{k+1} = \theta_t^k + \alpha_k \nabla_{\theta_t} l(\theta_t^k) \quad (9)$$

$$\nabla_{\theta_t} l(\theta_t) = \frac{\mathbb{E}_{\epsilon_t \sim P(\theta)} [\mathbf{S}(-\mathbf{J}(X, U)) \nabla_{\theta_t} \ln p(\epsilon_t; \theta_t)]}{\mathbb{E}_{P(\theta)} [\mathbf{S}(-\mathbf{J}(X, U))]}, \quad (10)$$

where α_k satisfies:

$$\alpha_k > 0 \quad \forall k, \quad \lim_{k \rightarrow \infty} \alpha_k = 0, \quad \sum_{k=0}^{\infty} \alpha_k = \infty, \quad (11)$$

in order to achieve asymptotic convergence [18]. If the shaping function is chosen to be the exponential function,

$$\mathbf{S}(y; \lambda) = \exp\left(\frac{1}{\lambda} y\right), \quad (12)$$

with λ being the inverse temperature, Eq. (10) becomes equivalent to the update law for information-theoretic MPPI [5],

$$\mathbf{u}_t^* = \mathbb{E}_{V \sim P(\theta)} [w(V)(\mathbf{v}_t)], \quad (13)$$

$$w(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} \mathbf{J}(X, V)\right), \quad (14)$$

where η is a regularization term, $V = \{\mathbf{v}_0, \dots, \mathbf{v}_{T-1}\}$ is a sampled control trajectory from the Gaussian control distribution $P(\theta)$, and w is the weight associated with V . As our distribution in the frequency domain is independent and Gaussian, we can change the t variable in Eqs. (9) and (10)

to n and subsume the iFFT transform inside of \mathbf{J} with the following equation, $\tilde{\mathbf{J}}(X, U) = \mathbf{J}(X, \Psi(U))$. This allows us to use the same update law to update the parameters of our frequency-domain sampling distribution,

$$\theta_n^{k+1} = \theta_n^k + \alpha_k \nabla_{\theta_n} l(\theta_n^k), \quad (15)$$

$$\nabla_{\theta_n} l(\theta_n) = \frac{\mathbb{E}_{\epsilon_n \sim P(\theta)} [\mathbf{S}(-\tilde{\mathbf{J}}(X, U)) \nabla_{\theta_n} \ln p(\epsilon_n; \theta_n)]}{\mathbb{E}_{P(\theta)} [\mathbf{S}(-\tilde{\mathbf{J}}(X, U))]} \quad (16)$$

For this paper, we choose to only update the mean of our control distribution, $\theta_n = \{\mu_n\}$, and leave the variance constant; variance updates would be derived the same way if desired. The gradient is estimated using Monte-Carlo sampling. Once the gradient is estimated, we choose to do the mean trajectory update in the time domain for computational convenience,

$$\mu_t^{k+1} = \Psi(\mu_N^{k+1}) = \Psi(\mu_N^k) + \alpha_k \Psi(\nabla_{\mu_N} l(\mu_N^k)), \quad (17)$$

where $\Psi(\mu_N^k)$ is equivalent to μ_t^k and

$$\Psi(\nabla_{\mu_N} l(\mu_N^k)) = \Psi\left(\sum_{m=0}^M w_m \mathcal{E}_N^m\right) = \sum_{m=0}^M w_m z^m(t), \quad (18)$$

with M being the total number of sample trajectories, μ_N referring to the collection of $\{\mu_n; n = 0, \dots, N\}$, \mathcal{E}_N^m refers to the m th sample trajectory's collection of frequency samples $\mathcal{E}_n^m = \{\epsilon_0^m, \epsilon_1^m, \dots, \epsilon_N^m\}$, and w_m is calculated using Eq. (14). The final algorithm for Frequency Sampling MPPI can be seen in Algorithm 1 with changes from the standard MPPI algorithm highlighted in blue.

IV. EXPERIMENTAL RESULTS

A. Off-road Vehicle Platform

We conducted experiments on a full-scale autonomous off-road modified Polaris Razer X, shown in Fig. 1. It is equipped with a sensor array mounted on top and an onboard computer using 4 RTX 3080s, 256 GB of RAM, and a Threadripper 3990x. The autonomy controls the vehicle using the power-assisted steering wheel actuator, a pump for brake pressure, and direct access to the throttle. The control inputs for the throttle and brake are combined to create a range of $[-1, 1]$ and the steering wheel control input is normalized to within $[-1, 1]$. Each control input has some form of delay from applying a signal and seeing an effect on the vehicle. For example, turning the steering wheel fully from right to left takes about two seconds. We learned the dynamics and modeling delays, $\mathbf{F}(\cdot, \cdot)$, using [19]. The slow response to controls is where the limitations of Gaussian sampling become clear. In order to achieve tight turning maneuvers like human drivers can, MPPI needs to sample large steering values in the same direction for multiple time steps in a row. While increasing the variance of the Gaussian distribution makes sampling larger values easier, it does nothing to address the fact that the samples need to have the same sign to actually make a sharp turn. Running several iterations can move the mean to one extreme but this is computationally expensive and slow-moving.

Algorithm 1: Frequency-based Sampling MPPI

Given: $\mathbf{F}(\cdot, \cdot)$, $q(\cdot)$, $\phi(\cdot)$ M , I , T , λ , σ , γ , ζ , f_{min} , α : System dynamics, running state cost, terminal cost, num. samples, num. iterations, time horizon, temperature, standard deviations, sampling exponents, sampling normalization term, minimum sampling frequency, update step size;

Input: \mathbf{x}_0 , \mathbf{U} : initial state, mean control sequence;

Output: \mathcal{U} : optimal control sequence

```

// Calculate Frequency range
1  $N \leftarrow \lfloor \frac{T}{2} \rfloor + 1$ ;
// Begin Cost sampling
2 for  $i \leftarrow 1$  to  $I$  do
3   for  $m \leftarrow 1$  to  $M$  do
4      $J^m \leftarrow 0$ ;
5      $\mathbf{x} \leftarrow \mathbf{x}_0$ ;
6     // Sample Frequency noise
7      $\mathcal{E}_N^m = (\epsilon_0^m \dots \epsilon_N^m)$ ,  $\epsilon_n^m \in \mathcal{N}(0, \zeta^{-1} (\max\{\frac{n}{N}, f_{min}\})^{-\gamma})$ ;
8     for  $t \leftarrow 0$  to  $T-1$  do
9        $\mathbf{z}^m(t) \leftarrow \Psi(\mathcal{E}_N^m)$ ;
10       $\mathbf{v}_t \leftarrow \mathbf{u}_t + \sigma \mathbf{z}^m(t)$ ;
11       $\mathbf{x} \leftarrow \mathbf{F}(\mathbf{x}, \mathbf{v}_t)$ ;
12       $J^m \leftarrow J^m + q(\mathbf{x})$ ;
13    // Compute trajectory weights
14     $\rho \leftarrow \min\{J^1, J^2, \dots, J^M\}$ ;
15     $\eta \leftarrow \sum_{m=1}^M \exp(-\frac{1}{\lambda} (J^m - \rho))$ ;
16    for  $m \leftarrow 1$  to  $M$  do
17       $w_m \leftarrow \frac{1}{\eta} \exp(-\frac{1}{\lambda} (J^m - \rho))$ ;
18    // Control update
19    for  $t \leftarrow 0$  to  $T-1$  do
20       $\mathcal{U}_t \leftarrow \mathbf{u}_t + \alpha \sum_{m=1}^M w_m \mathbf{z}^m(t)$ ;

```

In our experiments, both the Gaussian and Colored sampling MPPI algorithms are implemented in CUDA with $I = 3$ iterations, $M = 6144$ samples, $\lambda = 0.1$, $dt = 0.02s$, and $T = 250$ step horizon. The cost function for both algorithms is

$$\begin{aligned}
q(\mathbf{x}) = & \text{CostToGo}(\mathbf{x}) + \hat{d}(\text{WheelRisk}(\mathbf{x}) \\
& + \text{BodyRisk}(\mathbf{x}) + \text{Lethal}(\mathbf{x}) + \text{Rollover}(\mathbf{x}) \quad (19) \\
& + \text{Roll}(\mathbf{x}) + \text{Pitch}(\mathbf{x}) + \text{Speed}(\mathbf{x})),
\end{aligned}$$

where $\hat{d} = \mathbf{v}dt$ is the estimated distance traveled at a given timestep.

In Fig. 4, the vehicle is attempting to stay within the bounds of this zig-zag corridor which requires hard turns for appreciable periods of time to achieve. There is a human safety operator in the vehicle that would engage the manual override whenever the vehicle would exit the corridor. It is important to note that both controllers look to violate the boundaries of the corridor but this is due to drift in state estimation between the vehicle position and the corridor location. Gaussian sampling could not complete more than a single turn even when started in various locations whereas the Colored sampling only had one human intervention in its six attempts. The times and success rates are summarized in Table I. The computation times of the Colored MPPI averaged 26.846ms over these experiments while the vanilla MPPI algorithm averaged 24.839ms, showing there is minimal overhead to using the Colored approach on this hardware.

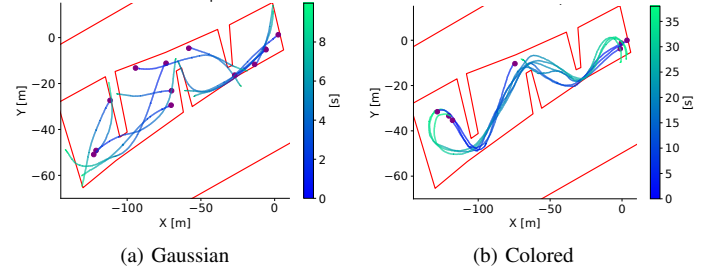


Fig. 4. Hardware experiments using Gaussian and Colored Sampling on the vehicle. These graphs show the vehicle trying to maneuver in a zig-zag corridor shown in red. The colors along the trajectories indicate the amount of time spent in autonomy before manual override/goal achieved with purple dots indicating the starting points. The Colored samples can go from one end of the corridor to the other while the Gaussian sampling struggles to make more than a single turn even when started in various locations.

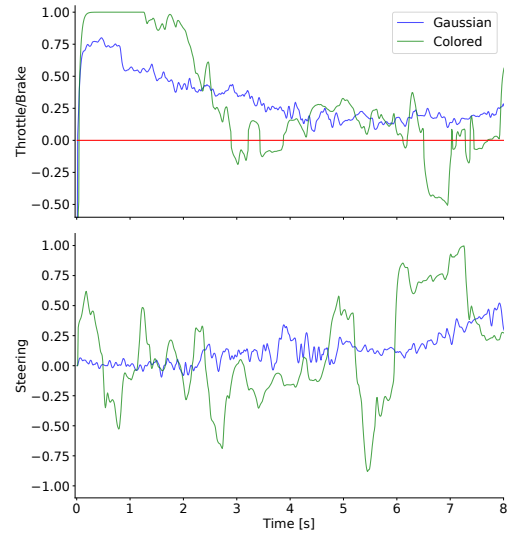


Fig. 5. Controls from the vehicle hardware attempts. Picture are the controls from the first eight seconds of the first attempt of each controller starting from the same location. We can see that the colored sampling technique is achieves smoother and larger throttle, brake, and steering commands. Throttle and brake are combined into a single graph with brake values being below the red line.

Furthermore, from the controls plotted in Fig. 5, we can further see that the Gaussian sampler can hit high steering angles of 0.5 but only slowly as the Model Predictive Control (MPC) nature of MPPI adjusts the mean of the steering trajectory in small steps. For this testing, the control standard deviations were both set to 0.8, the combined throttle and brake exponent was $\gamma_{throttle} = 10$, and the steering exponent was $\gamma_{steering} = 4$.

TABLE I
LAP EXPERIMENT SUMMARIES

Sampling	Avg Time[s]	Min Time[s]	\mathcal{S}_R [%]
Off-road HW Gaussian	N/A	N/A	0.00
Off-road HW Colored	38.33 \pm 3.32	35.990	83.33
Quad. SIM Gaussian	8.547 \pm 0.206	8.360	24.32
Quad. SIM Colored	8.691 \pm 0.457	7.695	42.86

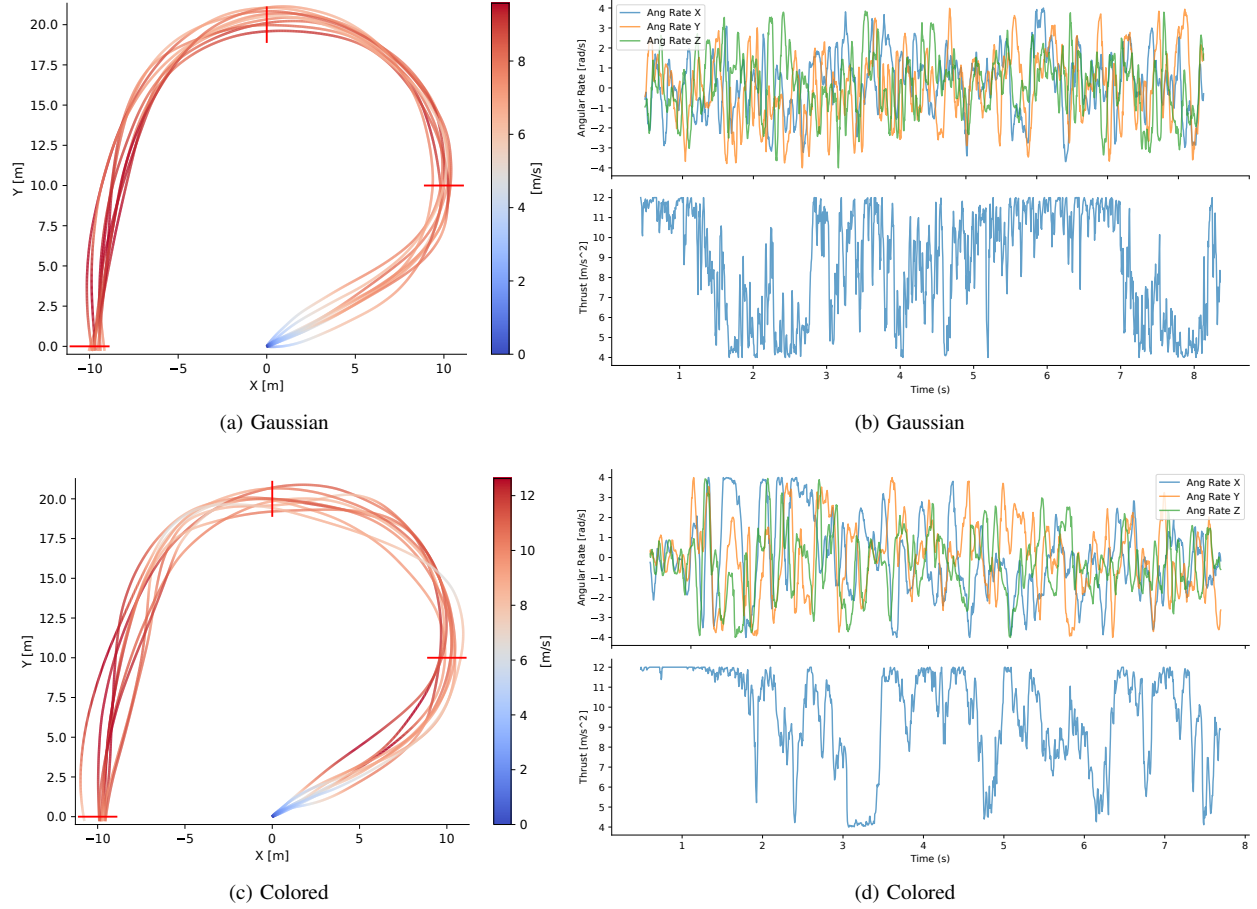


Fig. 6. Quadrotor flights using both types of sampling. Each controller is showing the first 9 successful attempts flying through this course. The red bars indicate gate positions that the quadrotor was expected to fly through. The left side graphs show the trajectories taken by the two controllers, colored according to the ground velocity of the quadrotor. On the right side, the controls that achieved the fastest lap for each controller are shown. Note that the throttle signal is much smoother for the Colored sampling when compared to Gaussian.

B. Simulated Quadrotor Results

We conducted additional experiments on a quadrotor platform in simulation. The simulation environment used is the Flightmare simulator [20], shown in Fig. 1. The dynamics function $\mathbf{F}(\mathbf{x}_t, \mathbf{u}_t)$ used within MPPI had a 13-dimensional state of $\{p_x, p_y, p_z, v_x, v_y, v_z, q_w, q_x, q_y, q_z, \omega_x, \omega_y, \omega_z\}$, where p is position [m], v is linear velocity [m s^{-1}], q is a quaternion representation of orientation, and ω are angular rates [rad s^{-1}]. It was assumed a low-level controller existed on the angular rates that could achieve a first order response of $\tau = 0.25$, and the control space was angular rates [rad s^{-1}] and thrust [m/s^2]. Both MPPI algorithms were run with $I = 2$ iterations, $M = 4096$ samples, $\lambda = 0.3$, $dt = 0.01\text{s}$, and $T = 150$. The quadrotor was to fly through gates known a-priori. The cost function was a combination of factors such as distance to next gate, maintaining height of 2m above the ground, tracking a desired velocity of 9m s^{-1} , minimizing the deviation of the thrust axis from vertical to maintain stable flight, and crash costs for hitting the gate,

$$\begin{aligned} q(\mathbf{x}) = & a_1 \text{Heading}(\mathbf{x}) + a_2 \text{Height}(\mathbf{x}) + a_3 \text{Speed}(\mathbf{x}) \\ & + a_4 \text{Stabilize}(\mathbf{x}) + a_5 \text{GateCrash}(\mathbf{x}) \\ & + a_6 \text{Waypoint}(\mathbf{x}) + a_7 \text{Path}(\mathbf{x}). \end{aligned} \quad (20)$$

When running MPPI with Gaussian sampling, we used standard deviations of $[0.3, 0.3, 0.3, 3.5]$ for the angular rates and thrust respectively. Note that the high thrust standard deviation is required for the quadrotor to adjust quickly when performing turns and reach quick lap times. As seen in Fig. 6a, the quadrotor successfully flies through the gates. Looking at the controls in Fig. 6b, we see that the controls end up being quite chattery, especially in thrust. This sort of chatter may lie outside of the control bandwidth of the quadrotor (i.e. going from 5m/s^2 to 11m/s^2 in 0.02s is not possible given the simulated motors) and can cause wear to physical motors over time.

When running MPPI with Colored sampling, we used standard deviations of $[0.2, 0.2, 0.2, 0.5]$ and exponents γ of $[0.01, 0.01, 0.01, 0.5]$ for the angular rates and thrust respectively based on best observed performance. Since the quadrotor is more reactive to control inputs than the hardware ground vehicle, lower exponents allow us to make sharp maneuvers while still reducing control chatter. While the lap times do not show much difference, we can see a difference in Fig. 6d of the controls of the fastest lap. Unlike the throttle of the Gaussian sampling case, we have lower frequency controls in throttle even when choosing a relatively small exponent of 0.5.

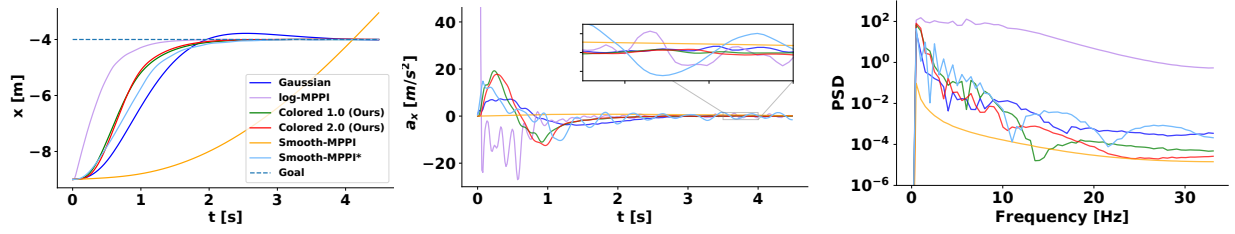


Fig. 7. State trajectories (**Left**), control trajectories (**Middle**), and PSD (**Right**) from the double integrator system with standard deviation set to 1.5. The PSD is shown for the first 2 seconds of the control trajectories to highlight the frequencies used to reach steady-state. Log-MPPI reaches the goal fastest but does so with lots of high frequency controls. The Colored methods using $\gamma = 1.0$ and 2.0 is able to react much quicker compared to Gaussian sampling by producing a much larger initial control spike and subsequent control dip to slow down to reach the goal without overshooting. Smooth-MPPI* does reach the goal sooner than Gaussian sampling but has larger control deviations around steady state. Finally, the Smooth-MPPI controller does not perform well with $\sigma = 1.5$, keeping the control pretty close to 0. The PSD shows that Colored control trajectories have lower contributions from high frequency components compared to Gaussian, Log-MPPI, and Smooth-MPPI*.

These are more likely to be within the control bandwidth of the motors and we only see a 0.15s average time loss from Colored sampling in Table I. The angular rates of Gaussian and Colored sampling do not see much difference but this is in line with the angular rates exponent, γ being set to 0.01. Looking at Fig. 6c, we can also see that the quadrotor achieved speeds of up to 12 ms^{-1} as it went towards the final gate compared to 10 ms^{-1} from the Gaussian laps.

Looking at Table I, we can see that Colored sampling had the fastest time through the course but could not reliably achieve that time. The controllers were both run until they achieved 9 valid attempts. We see that the Gaussian sampling took many more attempts to get 9 valid laps than the Colored sampling. Overall, the quadrotor system sees little performance hit for smoother control inputs.

C. Double Integrator Results

Finally, we conduct a test on a simple double integrator system. We tested six different controllers: MPPI with a Gaussian distribution, log-MPPI from [8], Smooth-MPPI from [9] as well as a modification described below, and MPPI with Colored sampling and colored exponents $\gamma = 1.0$ and 2.0 . The MPPI controllers had $M = 4096$ samples, $I = 1$ iteration, $dt = 0.015\text{s}$, $T = 65$, and $\lambda = 1$ for all sampling techniques. The systems started at -9 m and had a quadratic state cost of

$$q(\mathbf{x}) = 5(\mathbf{x}_0 + 4)^2 + 0.5\mathbf{x}_1^2. \quad (21)$$

We conducted comparisons of all methods at $\sigma = 0.5$, 1.5 , and 3.0 to see how adjustments to variance affect the performance of each controller. We reran all the controllers 1000 times to also see the variance caused by Monte-Carlo sampling which is reported in Table II. Note that for Smooth-MPPI, we saw poor performance using these standard deviations as the control would barely deviate from 0. We found that by dividing the provided standard deviation by dt , $\sigma^* = \frac{\sigma}{dt}$, performance fell in line with the other methods; we denote this change to the original method as Smooth-MPPI* and include both in our comparisons. In Fig. 7, we visualize the state and control trajectories that achieved the minimum cost for controllers using $\sigma = 1.5$. In addition, we also show the PSD of the first 2 seconds of each control trajectory to visualize how much each control frequency contributes to the trajectory as

it reaches the goal state. At this $\sigma = 1.5$, log-MPPI reaches the goal first but uses extremely large controls that would be impossible to achieve on hardware systems. Log-MPPI also is extremely jerky in attempting to slow down, as evidenced by high power concentration for large frequencies in the PSD. The Colored sampling distributions with exponents $\gamma = 1$ and 2 reach the goal next with smaller peak controls than log-MPPI. We see that the distribution with $\gamma = 2$ reaches the goal slightly faster and has lower power concentration at frequencies above 20Hz. They both have smaller control deviations from 0 at steady state compared to Smooth-MPPI* and log-MPPI. Smooth-MPPI* is the next to reach steady state but has oscillatory control behavior at steady state that is larger than all the other methods. Gaussian sampling ends up overshooting the goal and the control trajectory is not as reactive as log-MPPI or the Colored distributions. Finally, Smooth-MPPI as described in [9] barely reaches the goal in the time allotted. The control trajectory does not deviate from 0 by very much and has low power over all frequencies, showing that it is unable to respond quickly with this sampling standard deviation. Overall, Colored sampling allows the controller to be more reactive compared to Gaussian sampling while also being less oscillatory than both log-MPPI and Smooth-MPPI* at the same standard deviation.

We show the results across standard deviations of 0.5 , 1.5 , and 3.0 in Table II. The metric used in Table II for comparing the algorithms at various standard deviations is the accumulated cost of the states reached and we report the standard deviation of these costs over 1000 attempts. Depending on the standard deviation, log-MPPI, smooth-MPPI, and smooth-MPPI* can have orders of magnitude more variance in cost when tested multiple times compared to both Gaussian and Colored sampling. Increasing the control standard deviation improves all methods in general but the best controller changes as well. Our Colored sampling distributions maintain good relative performance, only losing to log-MPPI at higher standard deviations. In addition, we see in Fig. 7 that our method does not rely on control impulses or exhibit oscillatory behavior like controllers that do minimize the accumulated cost better. Interestingly, Smooth-MPPI becomes less consistent as the standard deviation increases but seems to have the best performance on average around $\sigma = 1.5$.

Looking at the Smooth-MPPI* results and Fig. 7, this is most likely due to the standard deviation choices being too small to properly excite the double integrator system. Finally, there is some computational overhead to running Colored sampling compared to other methods, but the increase in computation time at around 0.12ms seems to fall within the RTX 3080's performance variance and is only noticeable when the dynamics and cost function are so simple. While further tuning for each controller is possible to address any specific issue, we see that our proposed method performs well across a variety of standard deviations, both in terms of cost and control trajectory shape.

TABLE II
DOUBLE INTEGRATOR RESULTS

Std. Dev.	Sampling	Accumulated Cost	Calc. Time (ms)
0.5	Gaussian	27 919.0±33.1	0.26 ± 0.23
	Smooth-MPPI	61 330±17700	0.30 ± 0.17
	Smooth-MPPI*	21 900±19 20000	0.29 ± 0.23
	Log-MPPI	24 545.3±40.3	0.30 ± 0.12
	Colored 1.0	13 569.7±65.4	0.38 ± 0.16
	Colored 2.0	14 201.7±61.2	0.38 ± 0.16
1.5	Gaussian	13 815.5±64.6	0.26 ± 0.19
	Smooth-MPPI	44 400±32300	0.29 ± 0.15
	Smooth-MPPI*	11 446.3±71.1	0.29 ± 0.18
	Log-MPPI	7316±324	0.30 ± 0.19
	Colored 1.0	10 636.9±74.4	0.38 ± 0.18
	Colored 2.0	11 081.2±75.6	0.38 ± 0.22
3.0	Gaussian	10 815.1±75.7	0.26 ± 0.21
	Smooth-MPPI	68 090±75200	0.29 ± 0.13
	Smooth-MPPI*	10 505.5±75.6	0.29 ± 0.21
	Log-MPPI	6963±380	0.31 ± 0.16
	Colored 1.0	9191.7±69.6	0.38 ± 0.33
	Colored 2.0	9700.0±70.2	0.38 ± 0.25

V. CONCLUSIONS AND FUTURE WORK

In this work, we proposed use of a new sampling distribution with MPPI to facilitate smoother controls and a larger coverage of the state space. This Colored sampling distribution was shown to have nearly identical update laws when used in MPPI as the Gaussian distribution. This distribution has a tunable amount of higher frequency components which provides it the ability to be used across various dynamical systems. In our hardware results, our vehicle is unable to effectively maneuver when provided high-frequency samples from Gaussian distributions, reducing the reachable state space significantly. Meanwhile, our Colored sampling distribution demonstrated tight turns only achievable by sampling near the limits of the steering wheel for multiple timesteps in a row. On the simulated quadrotor, we see that by lowering our sampling exponents, we can achieve nearly similar performance to Gaussian-based MPPI while maintaining smoother throttle control. Finally, in the double integrator experiments, Colored sampling is able to achieve more extreme control trajectories than the Gaussian-based controller and produce lower frequency control trajectories than other methods such as log-MPPI and Smooth-MPPI. In the future, we plan to apply this sampling distribution on more advanced forms of MPPI such as Tube-MPPI and RMPPI as well as combine with other multi-hypothesis distributions such as GMM and Stein variational distributions.

VI. ACKNOWLEDGEMENTS

This research was developed with funding from the Jet Propulsion Laboratory, Defense Advanced Research Projects Agency, and the Office of Naval Research. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for public release and unlimited distribution.

REFERENCES

- [1] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*, ser. Modern analytic and computational methods in science and mathematics. American Elsevier Pub. Co., 1970. [Online]. Available: <https://cir.nii.ac.jp/crid/1130282270933519744> 1
- [2] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, vol. 4, pp. 1–51, 1995. 1
- [3] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and computing in applied probability*, vol. 1, pp. 127–190, 1999. [Online]. Available: <https://link.springer.com/article/10.1023/A:1010091220143> 1
- [4] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive Driving with Model Predictive Path Integral Control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440. [Online]. Available: <https://ieeexplore.ieee.org/document/7487277/> 1, 2
- [5] —, "Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8558663/> 1, 4
- [6] Z. Wang, O. So, K. Lee, and E. A. Theodorou, "Adaptive Risk Sensitive Model Predictive Control with Stochastic Search," in *Proceedings of the 3rd Conference on Learning for Dynamics and Control*. PMLR, 2021, pp. 510–522. [Online]. Available: <https://proceedings.mlr.press/v144/wang21b.html> 1, 4
- [7] J. Watson and J. Peters, "Inferring smooth control: Monte carlo posterior policy iteration with gaussian processes," in *Conference on Robot Learning*. PMLR, 2023, pp. 67–79. [Online]. Available: <https://proceedings.mlr.press/v205/watson23a/watson23a.pdf> 2
- [8] I. S. Mohamed, K. Yin, and L. Liu, "Autonomous Navigation of AGVs in Unknown Cluttered Environments: Log-MPPI Control Strategy," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 240–10 247, 2022. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9834098> 2, 7
- [9] T. Kim, G. Park, K. Kwak, J. Bae, and W. Lee, "Smooth Model Predictive Path Integral Control Without Smoothing," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 406–10 413, 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/9835021> 2, 7
- [10] Z. Wang, O. So, J. Gibson, B. Vlahov, M. S. Gandhi, G.-H. Liu, and E. A. Theodorou, "Variational Inference MPC using Tsallis Divergence," in *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, Apr. 2021. [Online]. Available: <https://www.roboticsproceedings.org/rss17/p073.html> 2
- [11] A. Lambert, F. Ramos, B. Boots, D. Fox, and A. Fishman, "Stein Variational Model Predictive Control," in *Proceedings of the 2020 Conference on Robot Learning*. PMLR, 2021, pp. 1278–1297. [Online]. Available: <https://proceedings.mlr.press/v155/lambert21a.html> 2
- [12] J. Sacks and B. Boots, "Learning Sampling Distributions for Model Predictive Control," in *Proceedings of The 6th Conference on Robot Learning*. PMLR, Mar. 2023, pp. 1733–1742. [Online]. Available: <https://proceedings.mlr.press/v205/sacks23a.html> 2
- [13] T. Power and D. Berenson, "Variational Inference MPC using Normalizing Flows and Out-of-Distribution Projection," in *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation, Jun. 2022. [Online]. Available: <http://www.roboticsproceedings.org/rss18/p027.pdf> 2
- [14] C. Pinneri, J. Achterhold, M. Rolínek, and G. Martius, "Sample-efficient Cross-Entropy Method for Real-time Planning," in *Proceedings of the 2020 Conference on Robot Learning*. PMLR, 2021, pp. 1049–1065. [Online]. Available: <https://proceedings.mlr.press/v155/pinneri21a.html> 2

- [15] Z. Zhang, J. Chen, Z. Chen, and W. Li, “Asynchronous Episodic Deep Deterministic Policy Gradient: Toward Continuous Control in Computationally Complex Environments,” *IEEE Transactions on Cybernetics*, vol. 51, no. 2, pp. 604–613, 2021. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/8946888> ³
- [16] J. Timmer and M. König, “On generating power law noise,” *Astronomy & Astrophysics*, vol. 300, pp. 707–710, 1995. [Online]. Available: <http://jeti.uni-freiburg.de/papers/timmer95.pdf> ³
- [17] R. Bracewell, “The Fourier Transform and its Applications.” McGraw-Hill, 2000, ch. 2. ³
- [18] E. Zhou and J. Hu, “Gradient-Based Adaptive Stochastic Search for Non-Differentiable Optimization,” *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1818–1832, 2014. [Online]. Available: <https://ieeexplore.ieee.org/document/6756948> ⁴
- [19] J. Gibson, B. Vlahov, D. Fan, P. Spieler, D. Pastor, A.-a. Aghamohammadi, and E. A. Theodorou, “A Multi-step Dynamics Modeling Framework For Autonomous Driving In Multiple Environments,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2023, pp. 7959–7965. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/10161330> ⁴
- [20] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, “Flightmare: A Flexible Quadrotor Simulator,” in *Proceedings of the 2020 Conference on Robot Learning*. PMLR, 2021, pp. 1147–1157. [Online]. Available: <https://proceedings.mlr.press/v155/song21a.html> ⁶