

Smooth Sampling-Based Model Predictive Control Using Deterministic Samples

Markus Walker* Marcel Reith-Braun*
Tai Hoang** Gerhard Neumann** Uwe D. Hanebeck*

* *Intelligent Sensor-Actuator-Systems*
Laboratory (ISAS), Institute for Anthropomatics and Robotics, Karlsruhe
Institute of Technology, Germany, {firstname}.{lastname}@kit.edu
** *Autonomous Learning*
Robots (ALR), Institute for Anthropomatics and Robotics, Karlsruhe
Institute of Technology, Germany, {firstname}.{lastname}@kit.edu

Abstract: Sampling-based model predictive control (MPC) is effective for nonlinear systems but often produces non-smooth control inputs due to random sampling. To address this issue, we extend the model predictive path integral (MPPI) framework with deterministic sampling and improvements from cross-entropy method (CEM)–MPC, such as iterative optimization, proposing deterministic sampling MPPI (dsMPPI). This combination leverages the exponential weighting of MPPI alongside the efficiency of deterministic samples. Experiments demonstrate that dsMPPI achieves smoother trajectories compared to state-of-the-art methods.

Keywords: Model predictive control, numerical methods for optimal control, deterministic sampling, cross-entropy method, model predictive path integral control.

1. INTRODUCTION

Sampling-based MPC methods have gained significant attention in recent years due to their ability to handle complex nonlinear systems and nonconvex cost functions. Similar to classical MPC, sampling-based MPC solves a finite-horizon optimal control problem (OCP) at each time step, and applies the first control input of the optimized control sequence to the system. Rather than relying on gradient-based optimization on the cost function, sampling-based MPC models the control input sequence as a parameterized discrete-time stochastic process and employs sampling-based optimization methods to iteratively improve the parameters. Specifically, samples of control input sequences are drawn from a proposal distribution, evaluated using the system dynamics and cost function, and then used to update the proposal distribution parameters. Using modern hardware, these steps can be performed efficiently in parallel. Popular sampling-based MPC methods include CEM–MPC (Chua et al., 2018; Pinneri et al., 2020) and MPPI (Williams et al., 2018; Bhardwaj et al., 2022).

A key challenge of sampling-based MPC methods is that they typically yield non-smooth control inputs (see, e.g., Fig. 1 for MPPI), as the proposal parameter update is approximated using a finite number of *random* samples, leading to noisy estimates that worsen for small sample counts. In real-world applications, this may cause problems, such as excessive actuator wear. E.g., noisy control inputs

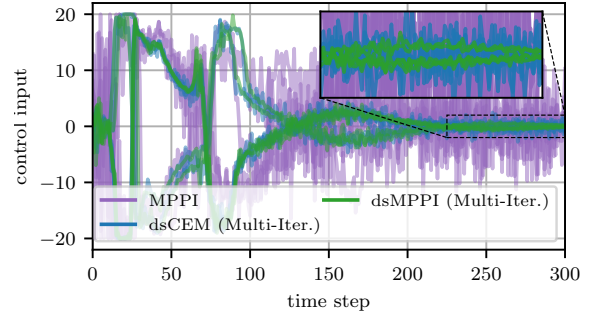


Fig. 1. Control input trajectories (five randomly initialized runs per method) for the cart-pole swing-up. The proposed dsMPPI yields the smoothest inputs.

may cause the torque of a motor to frequently switch between large positive and large negative values within a short time span, leading to rapid velocity reversals. Therefore, *smooth control inputs are desirable*. A common approach to addressing this issue is to apply low-pass filtering, to the optimized control inputs (Williams et al., 2018). However, this post-processing step leads to suboptimal performance, as it is not considered in the optimization.

In previous work, we proposed the *deterministic sampling CEM* (dsCEM) (Walker et al., 2026), which substitutes random sampling with deterministic samples generated by minimizing the modified Cramér–von Mises (CvM) distance (Hanebeck et al., 2009). This yields samples with low discrepancy, which cover the search space without clustering or leaving huge gaps, as random sampling does. This approach has been shown to significantly improve the smoothness of control inputs compared with standard CEM. However, dsCEM still relies on CEM’s elite set update

* This work is part of the German Research Foundation (DFG) AI Research Unit 5339 regarding the combination of physics-based simulation with AI-based methodologies for the fast maturation of manufacturing processes.

rule, which performs hard selection, weighing the N best samples equally, and disregarding all others. In contrast, MPPI uses a soft, exponential weighting scheme based on sample costs, enabling smoother updates. In this paper, we propose combining the benefits of deterministic sampling with the MPPI-style exponential weighting scheme. We term this approach *deterministic sampling MPPI* (dsMPPI) and show that it further improves the smoothness of control inputs compared to both MPPI and dsCEM, as illustrated in Fig. 1.

Contribution and Outline First, in Sec. 2, we provide a concise background on sampling-based MPC and review algorithmic improvements in Sec. 3. Then, in Sec. 4, we propose dsMPPI, a novel sampling-based MPC algorithm that utilizes state-of-the-art improvements from both CEM and MPPI literature, combined with deterministic sampling. Finally, in Sec. 5, we evaluate the performance of the proposed algorithm through extensive simulations.

Notation Vectors are denoted by underlined letters, e.g., $\underline{x}$, random variables by boldface letters, such as $\underline{\mathbf{x}}$, and matrices by boldface capitals, such as $\mathbf{A}$. The mean of a random variable is denoted by $\hat{\cdot}$, e.g., $\hat{\underline{x}}$, covariance matrices by $\mathbf{C}$, and diagonal matrices by $\text{diag}(\cdot)$. The indicator function $\mathbf{1}_A$ is 1 if A is true, and 0 otherwise.

2. BACKGROUND

We consider a discrete-time dynamical system given by

$$\underline{x}_{k+1} = \underline{a}_k(\underline{x}_k, \underline{u}_k), \quad (1)$$

and a finite-horizon OCP with cumulative cost

$$J_k = g_{N_H}(\underline{x}_{k+N_H}) + \sum_{n=k}^{k+N_H-1} g_n(\underline{x}_n, \underline{u}_n), \quad (2)$$

where k is the current time step, N_H is the prediction horizon, $\underline{x}_k \in \mathbb{R}^{d_x}$ is the system state, $\underline{u}_n \in \mathbb{R}^{d_u}$ is the control input, $g_n(\underline{x}_n, \underline{u}_n): \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \rightarrow \mathbb{R}$ is the stage cost at time step n , and $g_{N_H}(\underline{x}_{k+N_H}): \mathbb{R}^{d_x} \rightarrow \mathbb{R}$ is the terminal cost. Note that no further assumptions are made about the dynamics or cost functions, which can both be time-variant. The goal of the OCP is to find the optimal control sequence $\underline{u}_{k:k+N_H-1}^* = \underline{u}_k^*, \underline{u}_{k+1}^*, \dots, \underline{u}_{k+N_H-1}^*$ that minimizes the cumulative cost J_k subject to the system dynamics in (1). For brevity, we denote the flattened control sequence $\underline{u}_{k:k+N_H-1}$ as $\xi \in \mathbb{R}^{d_\xi}$ in the following, i.e., $\xi = [\underline{u}_k^\top, \underline{u}_{k+1}^\top, \dots, \underline{u}_{k+N_H-1}^\top]^\top \in \mathbb{R}^{N_H \cdot d_u}$, and the corresponding cumulative cost J_k as $J(\xi)$.

In sampling-based MPC methods, control sequences are drawn from a proposal distribution $f(\xi; \theta_j)$, parameterized by θ_j , where $j \in \{0, \dots, j_{\max} - 1\}$ denotes the iteration index and $j_{\max}$ is the number of iterations. The proposal is *iteratively* updated to generate control sequences with low costs that are near the optimal solution to the OCP. E.g., in the CEM proposed by Rubinstein et al. (2004), the proposal distribution is iteratively updated by solving

$$\begin{aligned} \theta_{j+1} &= \arg \min_{\theta'} D_{\text{KL}}(f_j^*(\xi) \| f(\xi; \theta')) \\ &= \arg \min_{\theta'} H(f_j^*(\xi), f(\xi; \theta')) \end{aligned}$$

where $D_{\text{KL}}(\cdot \| \cdot)$ is the Kullback–Leibler (KL) divergence, and $H(\cdot, \cdot)$ denotes the cross-entropy. The parameters are

updated by minimizing the KL divergence between an optimal *importance sampling* distribution f_j^* for iteration j and the proposal distribution $f(\cdot; \theta')$. This optimal importance sampling distribution $f_j^*(\xi)$, which minimizes the variance of the importance sampling estimator (Rubinstein et al., 2004), is given by $f_j^*(\xi) \propto \varphi(J(\xi)) f(\xi; \theta_j)$, where the weighting function $\varphi(J(\xi))$ assigns higher probability to control sequences with lower costs. Rewriting the cross-entropy term as an expectation w.r.t. the proposal distribution $f(\xi; \theta_j)$, expresses the update rule as

$$\theta_{j+1} = \arg \min_{\theta'} -\mathbb{E}_{f(\xi; \theta_j)} \{ \varphi(J(\xi)) \log f(\xi; \theta') \} \quad (3)$$

The expectation in (3) is approximated using samples $\{\xi^{(i)}\}_{i=1}^N$ drawn from $f(\xi; \theta_j)$, and solving for θ_{j+1} adapts the distribution to generate lower-cost samples in the next iteration. For Gaussian proposal distributions, where θ consists of the mean $\hat{\xi}$ and covariance matrix $\mathbf{C}$, the update (3) admits a closed-form solution given by the weighted sample moments

$$\begin{aligned} \hat{\xi}_{j+1} &= \sum_{i=1}^N \frac{w^{(i)}}{\eta} \xi^{(i)}, \\ \mathbf{C}_{j+1} &= \sum_{i=1}^N \frac{w^{(i)}}{\eta} (\xi^{(i)} - \hat{\xi}_{j+1})(\xi^{(i)} - \hat{\xi}_{j+1})^\top, \end{aligned} \quad (4)$$

where $w^{(i)} = \varphi(J(\xi^{(i)}))$ and $\eta = \sum_{i=1}^N w^{(i)}$ are the weights and the normalization constant, respectively.

A common special case is the use of $\varphi(\cdot) = \mathbf{1}_{J(\xi) \leq \gamma_j}$, that is, selecting an elite set of samples whose costs fall below a certain threshold γ_j . The update rule then reduces to sample moments computed only over the elite set, where each elite sample is weighted equally.

Using an exponential weighting $\varphi(J(\xi)) = \exp(-J(\xi)/\lambda)$ with inverse temperature $\lambda > 0$ relates the CEM to MPPI (Williams et al., 2018). Standard MPPI computes the control input $\underline{u}_k$ directly via weighted averaging according to (4) within only one iteration, i.e., without iteratively updating the proposal distribution. While MPPI connects to optimal control and information-theoretic principles (Williams et al., 2018), it imposes restrictive assumptions such as quadratic control costs. In contrast, the CEM formulation with exponential weights is more flexible and allows for arbitrary cost functions.

3. RELATED WORK

We now review relevant literature on sampling-based MPC. In this context, CEM typically denotes methods that employ an elite set, while MPPI refers to methods using an exponential weighting scheme. For the remainder of this paper, we adopt this convention.

3.1 Improvements to CEM–MPC

Standard improvements include warmstarting the proposal distribution based on the previous solution (Chua et al., 2018; Pinneri et al., 2020) and applying momentum smoothing to stabilize parameter updates (Rubinstein et al., 2004; Pinneri et al., 2020). To address the high dimensionality of control sequences, the covariance matrix is commonly

assumed to be diagonal, which simplifies estimation and sampling (Hafner et al., 2019). However, doing so neglects temporal correlations and sacrifices the smoothness of the control inputs.

In the *improved CEM* (iCEM), Pinneri et al. (2020) proposed a notable improvement that considers temporal correlations in the control sequence through random colored noise sampling with a power spectral density $\text{PSD}(f) \propto 1/f^\beta$, where f is the frequency and β controls the noise color. This allows for generating smoother control sequences while maintaining computational efficiency by updating only the marginal variances instead of the full covariance matrix. Additionally, iCEM uses a buffer to preserve a fraction of the previous elite samples, enhancing sample efficiency by maintaining effective solutions across iterations. Furthermore, the best solution found is returned after the MPC step.

In our previous work (Walker et al., 2026), we proposed the dsCEM, which uses deterministic samples generated offline by minimizing the modified CvM distance (Hanebeck et al., 2009) between a multivariate standard normal distribution and its sample approximation. These optimal samples are stored and transformed online to match the current proposal distribution. Deterministic sampling improves performance and sample efficiency compared to random sampling. However, using the same sample set for every iteration would result in poor exploration. To enhance exploration, we introduced variation schemes, such as selecting a different subset from a larger pool of precalculated samples for each iteration or applying random rotations to the stored samples. In addition, to capture temporal structure without incurring the computational cost of a full covariance matrix, we proposed updating only the marginal variances in each CEM iteration while using fixed time correlations.

3.2 Improvements to MPPI

For numerical stability, MPPI typically employs cost shifting relative to the minimum cost over all sampled trajectories (Williams et al., 2018). While standard MPPI performs a single weighted mean update ($j_{\max} = 1$) per MPC time step, iterative versions have been proposed. E.g., (Bhardwaj et al., 2022) integrated an MPPI-like exponential weighting into an iterative optimization scheme similar to CEM. A critical hyperparameter in MPPI is the inverse temperature λ , which controls the “sharpness” of the weights. Typically, λ is fixed or manually tuned. Recently, Pezzato et al. (2025) proposed a self-adaptation heuristic for λ that adjusts the temperature in each time step based on the costs of the sampled trajectories.

4. PROPOSED METHOD

We propose an improved sampling-based MPC algorithm that combines the strengths of CEM and MPPI with deterministic sampling. The key components of our proposed method, referred to as dsMPPI, are detailed below, and summarized in Alg. 1.

Iterative Update Similar to CEM-MPC, we use multiple iterations to update a proposal distribution for the control sequence. In each iteration j , the importance-weighted

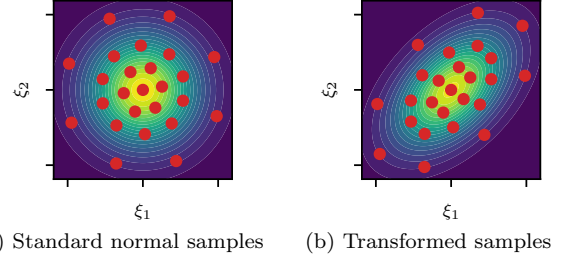


Fig. 2. Example of two-dimensional deterministic samples.

average known from MPPI is used, including a cost shift for numerical stability. That is, the weights are given by

$$w^{(i)} = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} \left(J(\xi^{(i)}) - \rho\right)\right), \quad (5)$$

where $\rho = \min\{J(\xi^{(i)})\}_{i=1}^N$ is the minimum cost over all trajectories, and η is the weight normalization constant. As in (Hafner et al., 2019), we assume a Gaussian proposal distribution for the control sequence, updating only the marginal variances to keep the dimensionality of the parameter space low. The update for the marginal variances $(\sigma'_j)^2$ of $\mathbf{C}'_j$ then becomes

$$(\sigma'_j)^2 = \sum_{i=1}^N w^{(i)} (\xi_j^{(i)} - \hat{\xi}'_j)^2, \quad (6)$$

where $(\cdot)^2$ denotes the element-wise square. Note that the quantities denoted by $\cdot'$ are intermediate quantities used for momentum smoothing.

Momentum Smoothing To stabilize the optimization and prevent premature convergence, we apply momentum smoothing to both the mean and covariance updates, as used in (Bhardwaj et al., 2022). Both the mean and covariance are updated via

$$\theta_{j+1} = \alpha \theta_j + (1 - \alpha) \theta'_j, \quad (7)$$

where $\theta'_j = (\hat{\xi}'_j, \mathbf{C}'_j)$ contains the weighted sample mean and covariance, and $\alpha \in [0, 1]$ is a momentum factor.

Adaptive Temperature To overcome the limitations of a fixed inverse temperature parameter λ , we use a heuristic adaptation rule. Similar to Pezzato et al. (2025), we use

$$\lambda_{j+1} = \begin{cases} 0.9 \lambda_j & \text{if } \eta > \eta_{\max} \\ 1.2 \lambda_j & \text{if } \eta < \eta_{\min} \\ \lambda_j & \text{otherwise} \end{cases}, \quad (8)$$

where η is the normalization constant from (5), and λ_j and λ_{j+1} are the inverse temperatures in iterations j and $j + 1$, respectively. Thresholds $\eta_{\min}$ and $\eta_{\max}$ are chosen empirically to ensure smooth behavior, and Pezzato et al. (2025) found that the range $[5, 10]$ performs well.

Deterministic Sampling with Variation Building upon our previous work (Walker et al., 2026), we incorporate deterministic samples to enhance the smoothness of the control inputs and improve sample efficiency. In each iteration, precomputed deterministic samples (Hanebeck et al., 2009) $\{\xi_{\text{SN}}^{(i)}\}$ from a isotropic standard normal $\xi \sim \mathcal{N}(\xi; 0, \mathbf{I})$ are transformed to match the current proposal distribution parameters $\theta_j = (\hat{\xi}_j, \mathbf{C}_j)$. The transformation for each sample $\xi_j^{(i)}$ is given by

$$\xi_j^{(i)} = \hat{\xi}_j + \mathbf{L}_j \xi_{\text{SN}}^{(i)}, \quad (9)$$

where $\mathbf{L}_j$ is the square root of $\mathbf{C}_j$ such that $\mathbf{C}_j = \mathbf{L}_j \mathbf{L}_j^\top$. Fig. 2 provides a visual intuition of this concept.

Since reusing the exact same deterministic samples in every iteration can lead to poor exploration, we implement two variation schemes to enhance diversity. A simple method is generating deterministic samples from an isotropic Gaussian with dimension $d_\xi \cdot j_{\max}$ and selecting different subsets of size d_ξ for each iteration j , as proposed in (Walker et al., 2026). We refer to this approach as the *multi-iteration* method. Since storing a large pool of deterministic samples can be memory-intensive, we propose a second variation scheme, referred to as the *permutation* method, that randomly permutes the dimensions of each deterministic sample in every iteration. E.g., in one iteration the dimension of the samples is interpreted as $[u_1, u_2, \dots, u_{d_u \cdot N_H}]$, whereas in the next iteration the dimensions are randomly permuted to, e.g., $[u_3, u_1, u_n, \dots]$. This approach requires storing only a single set of deterministic samples while still promoting exploration through dimension shuffling (this only requires a discrete random number generator). Furthermore, the standard normal samples remain optimal with respect to the modified CvM distance (Hanebeck et al., 2009) since an isotropic Gaussian is permutation-invariant.

Time Correlations For smooth control inputs, it is beneficial to introduce correlations between control inputs at different time steps. Following Walker et al. (2026), the covariance matrix is constructed as $\mathbf{C}_j = \text{diag}(\sigma_j) \mathbf{C}_\rho \text{diag}(\sigma_j)$, where $\mathbf{C}_\rho$ is the fixed time-correlation matrix and σ_j is the vector of marginal standard deviations. The transformation in (9) then uses $\mathbf{L}_j = \text{diag}(\sigma_j) \mathbf{A}_\rho$, where $\mathbf{A}_\rho$ is the matrix square root of $\mathbf{C}_\rho$. This approach allows for capturing temporal correlations while keeping the number of parameters manageable by only updating the marginal variances $(\sigma_j)^2$ at each iteration. Since the time-correlation structure is fixed, we calculate $\mathbf{A}_\rho$ offline to reduce computational overhead. As in Walker et al. (2026), we use a correlation structure derived from colored noise, following the power spectral density law, which has been shown to produce smooth control trajectories suitable for many robotic applications (Eberhard et al., 2023). The initial time correlation matrix $\mathbf{C}_t$ is derived from the power spectral density of colored noise using the Wiener–Khinchin theorem (Kay et al., 1993, pp. 576–578).

For multivariate control inputs, the overall correlation matrix $\mathbf{C}_\rho$ is constructed by $\mathbf{C}_\rho = \mathbf{C}_t \otimes \mathbf{C}_{\text{sp}}$, where $\otimes$ denotes the Kronecker product, and $\mathbf{C}_{\text{sp}}$ captures spatial correlations between different control input dimensions at the same time step. For simplicity, in this work we assume independent control input dimensions and set $\mathbf{C}_{\text{sp}} = \mathbf{I}$.

Further improvements To ensure that the sampled control inputs respect input bound constraints, clamping is applied. This can be seen as modified nonlinear system dynamics (Williams et al., 2018)

$$x_{k+1} = a_k(x_k, \text{clamp}(u_k, u_{\min}, u_{\max})) , \quad (10)$$

where $\text{clamp}(\cdot)$ restricts the control inputs to the specified bounds $u_{\min}$ and $u_{\max}$.

Additionally, we use a buffer of size N_{Buffer} to keep the best trajectories from the previous iteration, as in Pinneri

Algorithm 1: dsMPPI Step

Input: State x_k , initial parameters $\theta_0 = (\xi_0, \mathbf{C}_0)$, buffered samples from previous step

- 1 Warm start last proposal mean and buffered samples (11)
- 2 **for** $j \leftarrow 0$ **to** $j_{\max} - 1$ **do**
- 3 Transform deterministic samples (9)
- 4 Add saved samples from buffer
- 5 Trajectory shooting using (10) // **parallel**
- 6 Evaluate costs using (2) // **parallel**
- 7 Calculate weights $w^{(i)}$ using (5)
- 8 Calculate weighted sample moments (6)
- 9 Update proposal distribution using (7)
- 10 Update temperature parameter λ_j using (8)
- 11 Add N_{Buffer} best samples to buffer
- 12 **return** first control u_k^* from the best sequence

Table 1. Task-specific parameters.

Parameter	Cart-Pole Swing-Up	Truck Backer-Upper
Control limits	$u \in [-20, 20]$	$u_1, u_2 \in [-1, 1]$
Goal state x_g ($\tilde{x}_g$)	$[0, 0, 1, 0, 0]^\top$	0
State weights $\mathbf{Q}$	$\text{diag}([0.1, 0.1, 1, 0.1, 0.1])$	$\text{diag}([0.01, 0.5, 5, 0.01])$
State weights $\mathbf{Q}_{N_H}$	$\text{diag}([10, 0.1, 10, 0.1, 0.1])$	$\text{diag}([0.1, 1, 10, 0.1])$
Control weights $\mathbf{R}$	10^{-4}	$\text{diag}([10^{-3}, 5])$
Noise color β	0.25	1
Initial $\hat{\xi}_0$	0	0
Initial σ_0	$10 \cdot \mathbf{1}$	1
Iterations $j_{\max}$	3	3
Horizon N_H	30	15
Buffer size N_{Buffer}	3	3
Total time steps T	300	100

et al. (2020). This improves sample efficiency and allows returning the best found control input instead of the proposal mean at the end of the MPC step. Furthermore, as in Pinneri et al. (2020), we perform a warm start of the proposal distribution at each time step by shifting the mean and buffered trajectories based on the previous solution. Shifted sequences are given by

$$\xi = [u_{k-1,1:N_H-1}^\top, u_{k-1,N_H-1}^\top]^\top, \quad (11)$$

where $u_{k-1,n}$ is the control input at stage n within the horizon from the previous MPC step ($k-1$), and the last control input is repeated to maintain the horizon length.

5. EXPERIMENTS

In this section, we evaluate the performance of the proposed dsMPPI algorithm and the novel permutation variation scheme. We compare these contributions against our previous work, dsCEM (Walker et al., 2026), and standard MPPI (Williams et al., 2018). Additionally, we perform an ablation study using a random sampling variant of our proposed dsMPPI algorithm, denoted as *MPPI Iterative*, to isolate the benefits of deterministic sampling. We conduct experiments on the cart-pole swing-up and the truck backer-upper tasks. For both tasks, quadratic stage cost functions of the form $g_n(x_n, u_n) = (x_n - x_g)^\top \mathbf{Q} (x_n - x_g) + u_n^\top \mathbf{R} u_n$ are used, where x_g is the goal state, and $\mathbf{Q}$ and $\mathbf{R}$ are state and control weights, respectively. The terminal cost function is set to $g_{N_H}(x_{k+N_H}) = (x_{k+N_H} - x_g)^\top \mathbf{Q}_{N_H} (x_{k+N_H} - x_g)$. The task-specific parameters are summarized in Tab. 1.

We evaluate controller performance using cumulative cost and control input smoothness. The cumulative cost is the

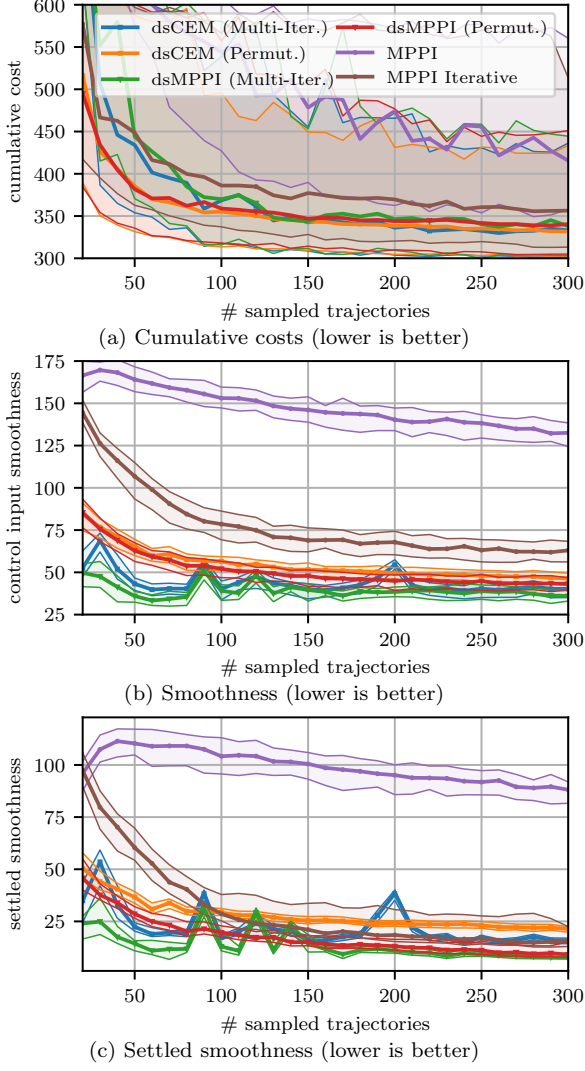


Fig. 3. Results for the cart-pole swing-up.

sum of stage costs $g_k(x_k, u_k)$ over the entire simulation. As in (Walker et al., 2026), the smoothness is measured using $\sum_{k=1}^{T-1} \|u_k - u_{k-1}\|^2$, where a lower value indicates a smoother trajectory. Since smoothness near the goal state is particularly important to avoid oscillations around the set point, we also report the smoothness measure restricted to the second half of the trajectory (time steps $\lceil T/2 \rceil$ to $T - 1$), denoted as *settled smoothness*. Each experiment uses sample sizes ranging from 20 to 300, and each experiment is repeated 100 times with different random seeds to randomly sample the initial states. The median and interquartile range per setting are reported.

Cart-Pole Swing-Up The task involves swinging a pendulum mounted on a cart from a downward to an upright position while balancing it there. We use the same experimental setup as Walker et al. (2026), where the state is $x = [x, \dot{x}, \phi, \dot{\phi}]^\top$, with cart position x , pole angle ϕ , and their respective (angular) velocities. The costs are evaluated using an augmented state vector $\tilde{x} = [x_1, x_2, \cos(x_3), \sin(x_3), x_4]$ to account for the periodicity of the angle x_3 . Results for the cart-pole swing-up are shown in Fig. 3. Control input trajectories for 300 samples are visualized in Fig. 1.

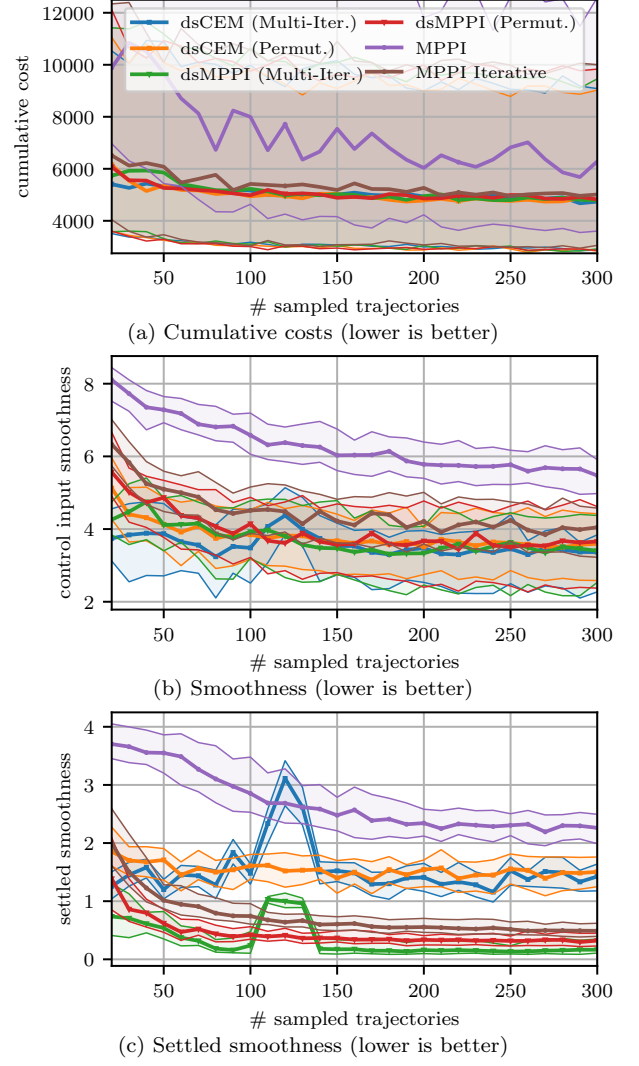


Fig. 4. Results for the truck backer-upper.

Truck Backer-Upper For the truck backer-upper (Schoenauer et al., 1994), the discrete-time dynamics reads

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{S,k+1} \\ \theta_{C,k+1} \end{bmatrix} = \begin{bmatrix} x_k - B \cdot \cos(\theta_{S,k}) \\ y_k - B \cdot \sin(\theta_{S,k}) \\ \theta_{S,k} - \arcsin\left(\frac{A \cdot \sin(\theta_{C,k} - \theta_{S,k})}{L_S}\right) \\ \theta_{C,k} + \arcsin\left(\frac{\tilde{u}_{2,k} \cdot \sin(\tilde{u}_{1,k})}{L_S + L_C}\right) \end{bmatrix}$$

where x_k, y_k are the rear axle coordinates in m and $\theta_{S,k}, \theta_{C,k}$ are the trailer and cab angles, respectively, relative to the x-axis. The control inputs are the steering angle $\tilde{u}_{1,k}$ and velocity $\tilde{u}_{2,k}$. The trailer and truck lengths are $L_S = 14$ m and $L_C = 6$ m, respectively, with $A = \tilde{u}_{2,k} \cos(\tilde{u}_{1,k})$ and $B = A(\theta_{C,k} - \theta_{S,k})$. A jackknife constraint clamps $|\theta_{C,k}|$ to 90° if the difference $|\theta_{S,k} - \theta_{C,k}|$ exceeds 90° . The optimization variables are the normalized inputs $u_{1,k}, u_{2,k} \in [-1, 1]$, scaled by the maximums 70° and 3 m per time step, respectively. Initial states are sampled uniformly from $x_0 \in [80, 100]$ m, $y_0 \in [-50, 50]$ m, and $\theta_{S,0}, \theta_{C,0} \in [-90, 90]^\circ$. The task is to back the trailer into the origin (0, 0) with both angles at 0° , i.e., $x_g = 0$. Results for the truck backer-upper are shown in Fig. 4.

Table 2. Computation times per control step.

Method	Cart-Pole Swing-Up ^a	Truck Backer-Upper ^a
MPPI	57.1 ± 8.4	10.7 ± 0.4
MPPI Iterative	166.6 ± 22.9	35.7 ± 2.0
dsMPPI Permut.	167.5 ± 27.4	37.4 ± 2.6
dsMPPI Multi-Iter.	166.7 ± 25.5	37.3 ± 2.7
dsCEM Permut.	166.1 ± 23.9	35.6 ± 2.8
dsCEM Multi-Iter.	167.0 ± 24.2	35.8 ± 3.2

^amean ± standard deviation in ms evaluated over 100 runs on a single core of an Intel Xeon Platinum 8358 processor.

Comparison of MPC Algorithms We first compare the proposed dsMPPI against MPPI Iterative, dsCEM, and standard MPPI. Standard MPPI performs significantly worse than the iterative methods in both tasks. Comparing the iterative methods, dsCEM generally achieves slightly lower cumulative costs than dsMPPI (Fig. 3a, Fig. 4a). However, dsMPPI still yields slightly lower cumulative costs than its random sampling counterpart MPPI Iterative in almost all settings, demonstrating the benefits of deterministic sampling.

In terms of control input smoothness, dsMPPI demonstrates superior performance. In the cart-pole swing-up, dsMPPI consistently yields smoother controls for large sample sizes than both dsCEM and MPPI Iterative (Fig. 3b). This advantage is even more pronounced in the settled smoothness (Fig. 3c), where dsMPPI outperforms all other methods. This is visually confirmed by the control input trajectories in Fig. 1. Similar results are obtained for the truck backer-upper (Fig. 4b, Fig. 4c), where dsMPPI consistently outperforms dsCEM, MPPI Iterative, and standard MPPI. This confirms that combining deterministic sampling with MPPI’s exponential weighting effectively mitigates chattering while maintaining competitive costs.

Comparison of Variation Schemes In terms of cumulative cost, the proposed permutation scheme slightly outperforms the multi-iteration scheme from (Walker et al., 2026). For the cart-pole swing-up (Fig. 3a), dsMPPI and dsCEM using the permutation scheme achieve lower costs than their multi-iteration counterparts, particularly for low sample sizes. A similar trend is observed in the truck backer-upper (Fig. 4a), where the permutation scheme yields competitive or lower costs.

Regarding control input smoothness, however, the multi-iteration scheme tends to produce smoother trajectories. In the cart-pole swing-up (Fig. 3b), the multi-iteration scheme consistently yields better smoothness for both dsMPPI and dsCEM. This trade-off is also evident in the truck backer-upper (Fig. 4b), where the multi-iteration scheme achieves the best smoothness for dsMPPI. Overall, the permutation scheme offers a cost advantage, while the multi-iteration scheme favors smoothness.

Computation Time Comparison Tab. 2 compares the computation times per control step, evaluated for 100 samples. Standard MPPI is the fastest method due to its non-iterative nature. Crucially, all iterative methods exhibit similar computation times, confirming that the proposed deterministic sampling scheme incurs no additional computational overhead.

6. CONCLUSION

In this paper, we proposed dsMPPI, a novel sampling-based MPC algorithm that integrates deterministic sampling with the exponential weighting scheme of MPPI. Additionally, we introduced a permutation-based variation scheme to enhance exploration with deterministic samples. Our extensive simulations on two nonlinear benchmark tasks demonstrate that dsMPPI effectively leverages the strengths of deterministic sampling and exponential weighting. It achieves significantly smoother control inputs compared to standard MPPI and dsCEM, while maintaining competitive cumulative costs. Furthermore, the proposed permutation scheme was shown to improve performance compared to the multi-iteration approach from dsCEM.

These characteristics make dsMPPI particularly promising for real-world applications. By generating smoother trajectories that do not require post-hoc filtering, dsMPPI can reduce mechanical stress and extend the lifespan of hardware components. Crucially, the proposed method incurs no additional online computational overhead compared to random sampling, making it a compelling upgrade for existing sampling-based MPC implementations.

REFERENCES

- Bhardwaj, M. et al. (2022). STORM: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Proc. 5th Conf. Robot Learn. (CoRL)*, 750–759.
- Chua, K. et al. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 4759–4770.
- Eberhard, O. et al. (2023). Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *Proc. 11th Int. Conf. Learn. Represent. (ICLR)*.
- Hafner, D. et al. (2019). Learning latent dynamics for planning from pixels. In *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2555–2565.
- Hanebeck, U.D. et al. (2009). Dirac mixture approximation of multivariate Gaussian densities. In *Proc. 2009 IEEE Conf. Decis. Control (CDC)*. Shanghai, China.
- Kay, S.M. (1993). *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, Inc.
- Pezato, C. et al. (2025). Sampling-based model predictive control leveraging parallelizable physics simulations. *IEEE Robot. Automat. Lett.*, 10(3), 2750–2757.
- Pinneri, C. et al. (2020). Sample-efficient cross-entropy method for real-time planning. In *Proc. Conf. Robot Learn. (CoRL)*, 1049–1065.
- Rubinstein, R.Y. et al. (2004). *The Cross-Entropy Method*. Information Science and Statistics. Springer New York.
- Schoenauer, M. et al. (1994). Neuro-genetic truck backer-upper controller. In *Proc. 1st IEEE Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, 720–723.
- Walker, M. et al. (2026). Sample-efficient and smooth cross-entropy method model predictive control using deterministic samples. In *Proc. 2026 Am. Control Conf. (ACC)*, 1–8. New Orleans, LA, USA.
- Williams, G. et al. (2018). Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Trans. Robot.*, 34(6), 1603–1622.