REDUCING THE SAMPLE COMPLEXITY OF SAMPLING-BASED CONTROL VIA HORIZON SPLITTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sampling-based methods such as MPPI, CMA-ES, and CEM are gaining traction for computing optimal control trajectories in robotics. We present sample complexity analysis, showing that zeroth-order methods require a number of samples exponential in the planning horizon, limiting their applicability to long-horizon tasks. Motivated by this analysis, we propose a rigorous horizon splitting algorithm to reduce the required sample complexity by sequentially solving subproblems. Validation on an optimal linear quadratic regulator (LQR) problem substantiates our theoretical predictions, and we show that our sample complexity bounds also model settings with no access to ground truth optimal solutions. On five robotics benchmarks, horizon splitting led to lower costs, especially over longer horizons, and typically outperformed the best sampling-based control setting without horizon splitting.

1 INTRODUCTION

Optimal control methods provide a mathematical framework for synthesizing feedforward trajectories and feedback controllers in dynamical systems. Direct optimization methods such as model predictive control are widely used, but their computational cost limits applicability to real-time computation for systems with long planning horizons or complex dynamics. Advancements in computational hardware have enabled a different paradigm of zeroth-order methods where optimality is assessed based on comparing a large number of candidate control sequences. In particular, for applications with differentiability limitations and low-quality gradients, these techniques, such as Model Predictive Path Integral (MPPI) (Williams et al., 2018), and Model-based Diffusion (MBD) (Pan et al., 2024), have gained increasing traction. However, obtaining sufficient samples to achieve desired behavior for high-dimensional and long-horizon tasks remains a core challenge, with limited understanding of sample complexity scaling laws.

In this paper, we derive scaling bounds for the sample complexity with respect to the probability of obtaining a control sequence that is close to the optimal solution, which we validate on an optimal Linear Quadratic Regulator (LQR). We then propose a divide-and-conquer algorithm that reduces the sampling complexity by splitting the horizon, which is grounded in our sample complexity results. Horizon splitting assesses the samples at intermediate time points and reweights them for the continued sampling until reaching the sampling time horizon. We apply horizon splitting to standard robotics benchmarks, and find that horizon splitting improves the performance of several sampling-based approaches, usually by 10% in terms of cost, and up to 90% in terms of task success.

This article presents three main contributions, in which we:

- Derive sample complexity bounds for certain sampling-based control algorithms;
- Present a horizon splitting algorithm using an approximate terminal cost and a sampling-based inner optimizer, which provably reduces the sample complexity of a class of sampling-based methods by an exponential factor;
- Validate our theoretical findings on an LQR experiment and demonstrate the applicability of our complexity bounds under relaxed assumptions as well as improved performance with horizon splitting for robotics benchmarks.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 Related Work

Sampling-based optimal control treats a finite action sequence as a black-box decision variable and improves a proposal distribution using sampled rollout costs. These optimizers are attractive when derivatives are unavailable, unreliable, or expensive. The Cross-Entropy Method (CEM) (Rubinstein, 1999) fits a proposal to elite samples through a KL projection, while CMA-ES (Hansen & Ostermeier, 2001) adapts a full Gaussian covariance and step size from selected search directions; MPPI (Williams et al., 2017) instead derives an exponential-cost-weighted update from path-integral control and importance sampling. MBD (Pan et al., 2024) iteratively refines trajectories using scores computed from the dynamics and cost. These methods reshape, reuse, or anneal proposals over complete trajectories. We propose a complementary approach that retains a generic inner sampler but organizes search across temporal segments, extending only prefixes selected by approximate costs.

The theoretical literature studies several different notions of sample efficiency. For path-integral control, Yoon et al. (2022) bound the multiplicative and additive error of the Monte Carlo control estimator at finite N and show severe horizon dependence of a variance upper bound for unstable linear systems. In contrast, CoVO-MPC (Yi et al., 2024) characterizes the infinite-sample contraction of MPPI on quadratic objectives and designs covariance schedules from the objective curvature. Most closely related is a concurrent work proposing Hybrid Feedback Sampling (Pan et al., 2026), which proves exponential-in-horizon sample requirements for isotropic open-loop MPPI on controllable unstable linear systems. Further, they remove this blow-up near an LQR optimum using temporally correlated feedback proposals. In contrast, we retain the underlying sampling optimizer and study a complementary mechanism: reducing the difficulty of sampling a near-optimal complete sequence by selecting and extending promising prefixes at intermediate horizons. Our analysis, therefore, focuses on finite-sample sequence-hitting probabilities under temporal decomposition.

Temporal decomposition has been previously explored for control methods. Direct multiple shooting introduces intermediate states and enforces continuity between separately integrated intervals (Bock & Plitt, 1984). More recently, the Trajectory Bundle Method uses sampled local interpolation inside sequential convex programming to enable derivative-free multiple shooting and explicit constraints (Tracy et al., 2026). In sampling-based planning, SMC Planning (Piché et al., 2019) sequentially grows trajectory particles and resamples their prefixes according to value-informed importance weights. Our scheme instead applies selection at segment boundaries, uses a hard cost band defined from an approximate cost-to-go, and extends surviving prefixes with an inner sampling-based optimizer. The closest algorithmic comparison is the concurrently proposed stochastic multiple-shooting method by Gupta & Moore (2026), which optimizes short CEM segments backward from a terminal set and connects them with locally identified time-varying LQR policies. In contrast, our horizon splitting approach grows dynamically feasible prefixes forward, permits a generic inner optimizer, and studies the resulting probability of sampling a near-optimal complete sequence.

3 Problem Formulation

Notation. We denote vectors by lower-case letters, matrices by capitalized letters, such as $x \in \mathbb{R}^n$, and sets, other than special sets like $\mathbb{R}$, by calligraphic letters. The state of the system is $x \in \mathbb{R}^n$, where n is the dimensionality of the state space $\mathcal{X}$. Similarly, $u \in \mathbb{R}^m$ denotes control inputs in the input space $\mathcal{U}$. The time index is indicated by a subscript, such as x_t, to denote the state at timestep t. Ordered sequences of time-indexed variables are denoted by range subscripts, like $1:t$, where omitted indices are taken to be 0 and ∞, respectively. Sets of many elements are denoted with curly braces, and superscripts denote members of the set. For example, $\{x^{(i)}\}_{i=1}^{N}$ is a collection of N elements, the k-th one is $x^{(k)}$. We use measures to denote probability distributions. For example, λ denotes standard Lebesgue measure. If we sample u from a measure μ, we write $u \sim \mu$. If $u \sim \mu$, then the probability that $u \in A$ is given by $\mu(A)$.

As with other sampling-based control methods, we are primarily interested in solving optimal control problems:

$$\begin{aligned} J(x, u_{1:}) &= \sum_{t=1}^{\infty} c(x_t, u_t) \\ \text{s.t.} \quad & x_{t+1} = f(x_t, u_t), x_1 = x. \end{aligned} \tag{1}$$

The goal is to minimize J with respect to $u_{1:}$ for a particular initial state x. The dynamics of the system are given by $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, and the stage cost, $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, measures the individual cost associated with each timestep. The cost-to-go associated with equation 1 is minimum achievable cost starting from state x:

$$V(x) = \min_{u_{1:}} J(x, u_{1:}). \tag{2}$$

Typically, it is infeasible to directly optimize J over the infinite horizon. Instead, we have access to a truncated cost

$$\begin{aligned} \tilde{J}(x, u_{1:T}) = &\sum_{t=1}^{T} c(x_t, u_t) + \Phi(x_{T+1}) \\ \text{s.t.} \quad &x_{t+1} = f(x_t, u_t), x_1 = x, \end{aligned} \tag{3}$$

with an approximate cost-to-go, $\Phi(x)$. We use the same symbol for all the functions $\tilde{J} : \mathcal{X} \times \mathcal{U}^T \to \mathbb{R}$, and implicitly define T based on the domain of the second argument.

This work addresses two questions with respect to equation 1. First, using a sampling-based method, how many samples are required to obtain a "satisfactory" solution? Second, having knowledge of the sample complexity of these approaches, can we improve on the state-of-the-art?

4 Sampling-based control

Typical sampling-based control methods operate in two iterative steps: defining the distribution to sample from, and evaluating samples to update the distribution (see algorithm 1). Typically, $\nu_1 = \delta_x$ is the Dirac delta distribution at x. Algorithms in this class include MPPI (Williams et al., 2018), CEM (Rubinstein, 1999), and MBD (Pan et al., 2024). An abstract overview of these algorithms can be found in (Kurtz & Burdick, 2026). Mainly, the choice and usage of μ, for example Gaussian distributions in MPPI, distinguishes the different approaches. For example, MPPI takes the first action of $\bar{u}_{1:T} = \mathbb{E}_{u\sim\mu}[u]$, and then repeats algorithm 1 from x_1. Meanwhile, Predictive Sampling (PS) samples from μ and takes the best sample.

4.1 Sample Complexity

In this section, we derive general complexity bounds for sampling-based control algorithms. Our primary focus is to determine the number of samples needed as a function of the optimization horizon. Consider sampling uniformly over a compact action space $\mathcal{U} \subset \mathbb{R}^m$, where m is the dimension of the action space. Fix an optimal action sequence $u^\star_{1:T} \in \mathcal{U}^T$. How many samples are required to ensure that, with high probability, at least one is near $u^\star_{1:T}$?

Definition 4.1. Define the δ-satisfactory neighborhood around $u^\star_{1:T}$ as

$$I^T_\delta(u^\star_{1:T}) = \left\{ u_{1:T} \in \mathcal{U}^T \,:\, \|u_{1:T} - u^\star_{1:T}\|_\infty \leq \frac{\delta}{2} \right\}. \tag{4}$$

Similarly, $u_{1:T}$ is a δ-satisfactory control sequence if $u_{1:T} \in I^T_\delta(u^\star_{1:T})$.

Algorithm 1 Generic Sample Predictive Control

Require: Approximate cost function $\tilde{J}$, initial state distribution ν_1, planning horizon T, number of iterations N_{iter}, and a number of samples N.

1: Initialize control distribution μ
2: **for** Iteration $k = 1, 2, \ldots, N_{\text{iter}}$ **do**
3: Sample initial conditions $\{x_1^{(i)}\}_{i=1}^N \sim \nu_1$
4: Sample control trajectories $\{u_{1:T}^{(i)}\}_{i=1}^N \sim \mu$ of length T
5: Compute costs $C^{(i)} \leftarrow \tilde{J}(x_1^{(i)}, u_{1:T}^{(i)})$ for $i = 1, \ldots, N$.
6: $\mu \leftarrow \texttt{Update}(\mu, \{u_{1:T}^{(i)}, C^{(i)}\}_{i=1}^N)$
7: **end for**
8: **return** μ

Remark 4.2. We use leading notation, also writing $I_\delta(u_t^\star) = \{u \in \mathcal{U} : \|u - u_t^\star\|_\infty \leq \delta/2\}$. Indeed, since $\|u_{1:T} - u_{1:T}^\star\|_\infty \leq \alpha$ if, and only if, $\|u_t - u_t^\star\|_\infty \leq \alpha$ for all $1 \leq t \leq T$, we have

$$I_\delta^T(u_{1:T}^\star) = I_\delta(u_1^\star) \times \cdots \times I_\delta(u_T^\star). \tag{5}$$

We are particularly interested in cases where we sample control inputs independently between timesteps, as done in most methods, e.g. MPPI (Williams et al., 2018), PS (Howell et al., 2022), and MBD (Pan et al., 2024).

Definition 4.3 (Time-factorized measure)**.** We call a sampling distribution, μ on $\mathcal{U}^T$ time-factorized if there exists probability measures $\mu_1, \ldots, \mu_T$, each on $\mathcal{U}$, such that

$$\mu(A_1 \times A_2 \times \cdots \times A_T) = \prod_{t=1}^{T} \mu_t(A_t). \tag{6}$$

for all Borel measurable sets $A_1, \ldots, A_T \subset \mathcal{U}$.

In these cases, the timestep-wise probability measures of the δ-satisfactory intervals also decompose, given equation 5.

Assumption 4.4 (Time-factorized sampling)**.** Action sequences are sampled i.i.d. from a time-factorized measure $\mu = \mu_1 \otimes \cdots \otimes \mu_T$ on $\mathcal{U}^T$. In Algorithm 2, this applies to the actions $u^{(i)}_{T_{k-1}:T_k}$ drawn in each segment, which are also sampled independently of the resampled prefixes $u^{(i)}_{1:T_{k-1}}$.

Sampling-based algorithms rely on sampling control sequences randomly to synthesize distributions of increasing quality. They do so without a priori knowledge of $u^\star$, which we formalize now.

Assumption 4.5. There is a number $p_\delta \in (0, 1)$ such that for all $T \geq 1$,

$$p_\delta \geq \sup_{1 \leq t \leq T} \sup_{u \in \mathcal{U}} \mu_t(I_\delta^T(u)).$$

Loosely speaking, the term corresponds to the probability of sampling in the easiest $I_\delta(u)$ over u. Notably, it holds for MPPI, where μ is a Gaussian distribution with covariance $\sigma^2 I$ and $p_\delta = \left(\frac{\delta}{\sigma\sqrt{2\pi}}\right)^m$. It also holds for uniform sampling, with $p_\delta = \delta^m / \lambda(\mathcal{U})$. The exact quantity depends on the algorithm and its parameters, but we give a general lower bound according to the following proposition. In brief, finding good action sequences requires a number of samples exponential in the planning horizon.

Proposition 4.6. *Let Assumption 4.4 and Assumption 4.5 hold. Let* $u^{(1)}_{1:T}, \ldots, u^{(N)}_{1:T} \sim \mu$*. If*

$$\Pr\Big(\textit{there is some } i \in \{1, \ldots, N\} \textit{ s.t. } u^{(i)}_{1:T} \in I_\delta^T(u^\star_{1:T})\Big) \geq 1 - \varepsilon, \tag{7}$$

then we must have

$$N \geq \left(\frac{1}{p_\delta^T} - 1\right) \log\left(\frac{1}{\varepsilon}\right). \tag{8}$$

The proof is in Appendix A.1. In practice, p_δ is not very close to 1. Thus, optimizing over a horizon of length T may require prohibitively many samples. If we could break the horizon T into K segments and do this sampling procedure K times, we could expect, loosely, to need a number of samples about $N \propto K p_\delta^{-T/K}$. This hinges on being able to determine near-optimality independently per-segment, in which case we perform a sampling-based procedure independently for each segment.

4.2 Horizon Splitting

The core idea of our algorithm is performing a "divide and conquer" procedure, reminiscent of standard sorting algorithms, using the terminal cost to reduce the computational complexity. Specifically, we require that the terminal cost has a bounded error.

Assumption 4.7 (Bounded terminal-cost error)**.** There is a known $\Delta \geq 0$ such that

$$\|V - \Phi\|_\infty = \sup_{x \in \mathcal{X}} |V(x) - \Phi(x)| \leq \Delta. \tag{9}$$

We split the optimization horizon T into K sub-horizons, each of length T/K. We then sample from the first sub-horizon and measure the cost of each control sequence. To sample from the next sub-horizon, we pick the initial conditions (or, equivalently, initial control sequences) that had promising costs in the previous horizon. We repeat this procedure sequentially for all sub-horizons.

Algorithm 2 A naive split-horizon sampling

Require: Approximate cost function $\tilde{J}$ with terminal cost Φ, planning horizon T, number of segments K, number of samples N, inner sampling-based optimizer `Optimize`, initial state x_1, number of inner iterations N_{iter}, cost error Δ.

1: Initialize $\nu_1 \leftarrow \delta_{x_1}$
2: Solve $T = Kp + r$ for $p \in \mathbb{N}, 0 \leq r < K$.
3: $\ell_k \leftarrow p + 1$ for $k = 1, \ldots, r-1$
4: $\ell_k \leftarrow p$ for $k = r, \ldots, K$
5: $T_k \leftarrow \sum_{j=1}^{k} \ell_j$.
6: $T_0 \leftarrow 1$.
7: **for** time segment $k = 1, 2, \ldots, K$ **do**
8: $\quad \mu_k, \{x_{T_k}^{(i)}, u_{T_{k-1}:T_k}^{(i)}, C_k^{(i)}\}_{i=1}^{N} \leftarrow$ `Optimize`$(\tilde{J}, \nu_k, \ell_k, N_{\text{iter}}, N)$
$\quad$ //NOTE: $C_k^{(i)} = \sum_{t=T_k}^{T_{k+1}} c(x_t^{(i)}, u_t^{(i)})$, without $\Phi(x_{T_k}^{(i)})$.
9: $\quad C^{(i)} \leftarrow \sum_{t=1}^{k} C_t^{(i)} + \Phi(x_{T_k}^{(i)})$ for all i.
10: $\quad C_{\min} \leftarrow \min_i C^{(i)}$
11: $\quad B_\Delta \leftarrow \{i \in 1, \ldots, N \,:\, C^{(i)} \leq C_{\min} + 2\Delta\}$
12: $\quad$ `indices` $\leftarrow$ Sample N indices uniformly from B_Δ, with repetition.
13: $\quad \mu_{1:k} \leftarrow \left[u_{1:T_k}^{(i)} \,:\, i \in \texttt{indices}\right]$
14: $\quad \nu_{k+1} \leftarrow \left[x_{T_k}^{(i)} \,:\, i \in \texttt{indices}\right]$
15: **end for**
16: **return** $\mu_{1:K}$

Algorithm 2 details this approach. Specifically, lines 9–13 are describing a particular reweighting scheme when sampling for the next sub-horizon.After observing the newest costs, we pick only the action sequences and associated initial conditions (which correspond to prior action sequences) which are potentially optimal (see figure 1). Notice that it is always true that $x_{T_k}^{(i)} = f(x_0, u_{1:T_k}^{(i)})$, since the distributions are always simultaneously reindexed, for all timesteps up to the current T_k. Additionally, the distributions are just the sets of samples drawn by `Optimize` in previous sub-horizons. Note that the whole sequence of actions $u_{1:T_k}^{(i)}$ is an argument to f. Since we consider deterministic dynamics, this means that we apply each input in sequence through the dynamics.

To analyze the algorithm and its sample complexity, we leverage the results from the previous section. In particular, under Assumption 4.7, we can prune trajectories before rolling them out further. Suppose that we have sampled action sequences for the first k sub-horizons. Following the notation and parameters in algorithm 2, we have N action sequences of length T_k with their costs. To sample the next horizon, we must resample from all potentially optimal sequences.

Definition 4.8. Given $\delta, \Delta > 0$, the (δ, Δ, T_k) near-optimal set of J, with optimizer $u_{1:T}^\star$ and initial state x_1 is defined by

$$\Omega_\Delta^{T_k}(x_1) := \left\{ u \in \mathcal{U}^{T_k} \,:\, |\tilde{J}(x_1, u_{1:T_k}) - \tilde{J}(x_1, u_{1:T_k}^\star)| \leq 2\Delta \right\}, \tag{10}$$

$$B_\Delta^{T_k}(x_1) := \bigcup_{u \in \Omega_\Delta^{T_k}(x_1)} I_\delta^{T_k}(u). \tag{11}$$

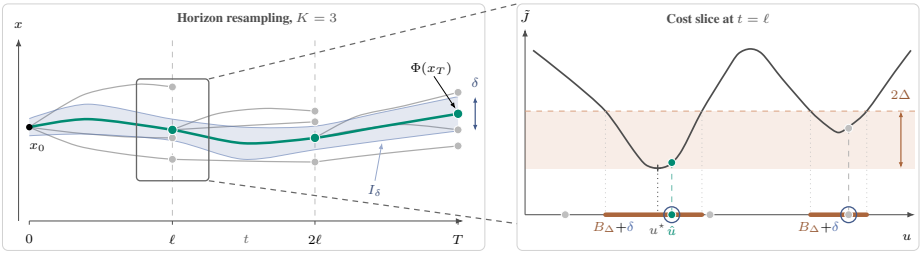


Figure 1: Horizon splitting. At each boundary (left), promising prefixes are resampled and extended from their endpoint states. The blue band marks the target neighborhood I_δ around the reference path. Selection is cost-based (right): the 2Δ threshold can retain prefixes in separated regions B_Δ, including ones outside I_δ.

Many of the sequences seen up to T_k will not be members of $B_\Delta^{T_k}(x_1)$. Putting this all together – the probability of correctly sampling from both $I_\delta^{T_k} \subset B_\Delta^{T_k}(x_1)$ and $I_\delta^{T_k:T_{k+1}}$ – we arrive at our main theorem. Following the notation we have employed throughout, for a time-factorized measure $\mu = \mu_1 \otimes \cdots \otimes \mu_T$ (Assumption 4.4), we use the notation $\mu_{t_1:t_2}$ to mean $\mu_{t_1} \otimes \cdots \otimes \mu_{t_2}$ for $1 \le t_1 \le t_2 \le T$.

Theorem 4.9. *Let Assumptions 4.4, 4.5, and 4.7 hold, and suppose we sample according to an idealized version of algorithm 2 that samples prefix control sequences from the true population conditional distribution. If*

$$\Pr\Big(\textit{there is some } i \in \{1, \dots, N\} \textit{ s.t. } u_{1:T}^{(i)} \in I_\delta^T(u_{1:T}^\star)\Big) \ge 1 - \varepsilon, \tag{12}$$

then we must have

$$N \ge p_\delta^{-\frac{T}{K}} \log\left(\frac{1}{\varepsilon}\right) \sum_{k=0}^{K-1} \tilde{p}_\delta^{-k\frac{T}{K}} \left(1 - \tilde{p}_\delta^{k\frac{T}{K}} p_\delta^{\frac{T}{K}}\right), \tag{13}$$

for some $\tilde{p}_\delta \in [p_\delta, 1]$.

We make a few remarks. First, algorithm 2 approximates the process described in the previous theorem using particles. Second, as $\Delta \to 0$, then $\tilde{p}_\delta \to 1$. This corresponds to having perfect knowledge of the cost-to-go $V(x)$. The sum in equation 13 becomes $K(1 - p_\delta^{T/K})$, and the overall complexity tends to $N = K p_\delta^{-\frac{T}{K}}(1 - p_\delta^{\frac{T}{K}}) \log(1/\varepsilon)$, as expected. If, on the other hand, we have no terminal cost, and $\Delta = \infty$, then $\tilde{p}_\delta = p_\delta$, and resampling only incurs additional samples. In any case, this model has some predictive power, which we investigate further in section 5. Finally, we note that our analysis assumes that the controller executes one of its samples, which is not true for all sampling-based methods. For example, MPPI iteratively refines its distribution and executes the final mean. In section 5.2, we empirically show that our model nonetheless predicts the behavior of these methods, but leave a more comprehensive investigation to future work.

5 Experiments

Our goals for this section are two-fold: first, we show that the relationship in equation 8 holds in the LQR setting, where we have access to ground truth. And, second, we show the reduced sample complexity in complex settings, for example in a high-dimensional humanoid stand-up task.

5.1 Linear Quadratic Regulator

In brief, the LQR problem is the optimal control problem with quadratic cost and linear dynamics:

$$\begin{aligned} V(x) = \min_{u_0, u_1, \dots} \quad & \sum_{t=1}^{\infty} x_t^\mathsf{T} Q x_t + u_t^\mathsf{T} R u_t \\ \text{s.t.} \quad & x_{t+1} = A x_t + B u_t, \quad t = 0, 1, \dots \\ & x_0 = x, \end{aligned} \tag{14}$$

where we consider the infinite-horizon problem to avoid time-dependent optimal control laws. It is well known that

$$V(x) = x^\mathsf{T} P x, \quad \text{and} \quad u^\star(x) = -Kx, \tag{15}$$

for some matrices $P \in \mathbb{R}^{n\times n}, K \in \mathbb{R}^{m\times n}$. We model the motion of a particle with a 2D integrator with $Q = I_4, R = 0.05 \cdot I_2$ implemented in Mujoco (Todorov et al., 2012) as a point mass with slide actuators (see Appendix A.2 for details). Using the previous equations, we can compute the exact cost-to-go, perturb it to simulate different values of Δ, and investigate their effects on the required number of samples. We also have access to the optimal control inputs, and measure the difference between the designed input via sampling, and the true optimal cost.

To validate our theoretical sampling bounds, we varied the noise on the terminal cost, which affects the $\tilde{p}_\delta$ parameter in Theorem 4.9. In particular, we use the terminal cost $\Phi(x) = V(x) + w(x)$ with $w(x) \sim U(-\Delta, \Delta)$ i.i.d, in which case the approximation certainly holds. Our success criterion is if the optimal plan generated by sampling is close to the optimal $u^\star$ in equation 15. Specifically, the success fraction $1-\varepsilon$ is estimated by counting the number of plans for which $\|u_{1:T} - u^\star_{1:T}\|_\infty \leq \delta/2$, where we set $\delta = 0.32$. The δ mainly serves visualization purposes, and is small compared to the action limits $[-1, 1]$ in each dimension and $T = 12$. Our results are presented in figure 2. The primary validation of our model is that $\tilde{p}_\delta$ is indeed relatively constant for a single Δ.

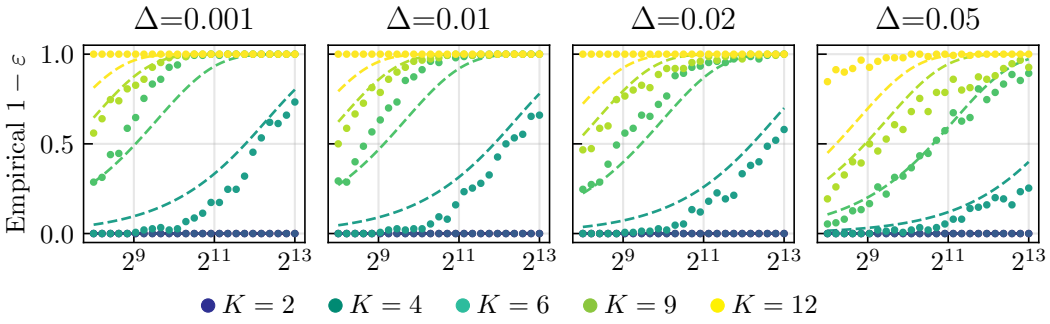


Figure 2: Empirical success fraction as a function of the number of samples, on the x axis, and for different noise levels Δ. The success fraction counts the fraction of control sequences within $\delta/2$ of the optimal control sequence in infinity-norm, The dashed lines correspond to the prediction given by Theorem 4.9, where we fit $\tilde{p}_\delta$ using nonlinear least squares. The dot and line colors represent the number of segments K used for those runs.

5.2 Validation on Control Tasks

As a next step, we apply our model to more complicated settings where we do not have access to ground truth, exemplified with humanoid stand-up as a standard robotics benchmark. We aim to assert that the number of samples required is indeed exponential in the planning horizon T. Previously, we leveraged a relationship between ε and N using the ground optimal action sequence $u^\star_{1:T}$. Now, we find the relation between N and δ. If we assume uniform sampling, then $p_\delta = \delta^m / \lambda(\mathcal{U})$. Reorganization of equation 13 yields the estimate

$$\|u^{\text{best}}_{1:T} - u^\star_{1:T}\|_\infty \sim \delta \sim \lambda(\mathcal{U})^{\frac{1}{m}} \left[\frac{1}{N} \log\left(\frac{1}{\varepsilon}\right) S(\delta, T, K)\right]^{\frac{K}{mT}}, \tag{16}$$

where u^{best} is the synthesized input sequence, and $S(\delta, T, K)$ denotes the sum in equation 13.

In practice, we can only observe the cost $\tilde{J}$ for the time horizon. To estimate our model, we assume that J is Lipschitz continuous with constant L. That is, for all u_1, u_2, there is an $L < \infty$ such that

$$|J(u_1) - J(u_2)| \leq L\|u_1 - u_2\|_\infty. \tag{17}$$

This implies that

$$\begin{aligned} |\tilde{J}(u^{\text{best}}_{1:T}) - \tilde{J}(u^\star_{1:T})| &\leq J(u^{\text{best}}_{1:T}) - J(u^\star_{1:T}) + 2\Delta \\ &\leq 2\Delta + L\|u^{\text{best}}_{1:T} - u^\star_{1:T}\|_\infty = 2\Delta + c\left[\frac{1}{N} \log\left(\frac{1}{\varepsilon}\right) S(\delta, T, K)\right]^{\frac{K}{mT}}, \end{aligned} \tag{18}$$

where we have subsumed constants into c. With equation 18 we can check if our model agrees with experiment in complex tasks. Specifically, we measure the empirical cost and compare it to the fitted approximation

$$\tilde{J}(u_{1:T}^{\text{best}}) = \alpha + \beta \left[\frac{1}{N} \log \left(\frac{1}{\gamma} \right) S(\delta, T, K) \right]^{\frac{K}{mT}}, \tag{19}$$

where the constants α, β, γ, and $\tilde{p}_\delta$ are fit once per task. Briefly, α subsumes $J(u^\star) + 2\Delta$, β subsumes L and c, and γ contains ε. Note that the curve changes concavity at some point, and eventually asymptotically reaches $\alpha + \beta \tilde{p}_\delta^{-K/m}$ as $T \to \infty$. Additionally, the S term drops out when $K = 1$, so we start with this regime.

We test this model on the humanoid stand-up task and present results for different horizon splits in figure 3. Importantly, we observe that our model and the empirical results qualitatively match well. Additionally, comparing the top row with the bottom shows that horizon splitting improves the cost with the same sample budget. To reduce fitting bias, we fit these 3 parameters on a large collection of variables. We vary T, N, k over 348 configurations.

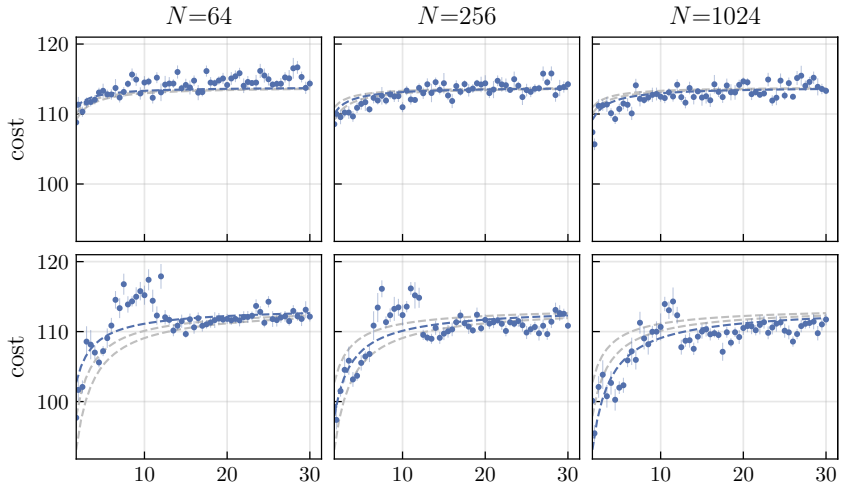


Figure 3: Comparison of empirical cost as a function of time horizon for different sample set sizes N of the humanoid stand-up task. The top row sets $K = 1$, while the bottom row sets $K = 6$. In each panel, the corresponding model is highlighted, and the other models for different N are grayed out for reference.

5.3 Comparison to baselines

Finally, to more comprehensively validate that our approach improves current sampling strategies, we investigate "monolithic" sampling methods. Note that we leave comprehensive performance benchmarking with respect to hierarchical planning methods or other shooting methods to future work, as the focus is to validate our theoretical analysis. For each task, we use the tuned parameter settings for the Hydrax repository examples as baselines (Kurtz, 2024), and then vary the planning horizon and number of sub-horizons. The experiment parameters are in the Appendix, Table 2.

We present our results in figure 4, which shows that horizon splitting typically improves performance. Specifically, for four of five tasks, our method achieves the lowest overall cost. We note that the first configuration in each plot corresponds to the highly-tuned, no-splitting, sampling controllers. Afterwards we only change the planning horizon and see if horizon splitting helps. As the planning horizon lengthens, our approach shows additional benefit, with the greatest improvement of 19% on walker. We note that on the humanoid task, we did not reduce the cost, but did find that the humanoid stood more often upright. Specifically, MPPI had a torso height exceeding $0.60\,\text{m}$ in no episodes, whereas MPPI with horizon splitting achieved a torso height of $0.71\,\text{m}$ in 90% of the episodes. Table 1 presents a quantitative comparison between the best setting with and without horizon splitting. The best setting without horizon splitting is always the default setting (i.e., first

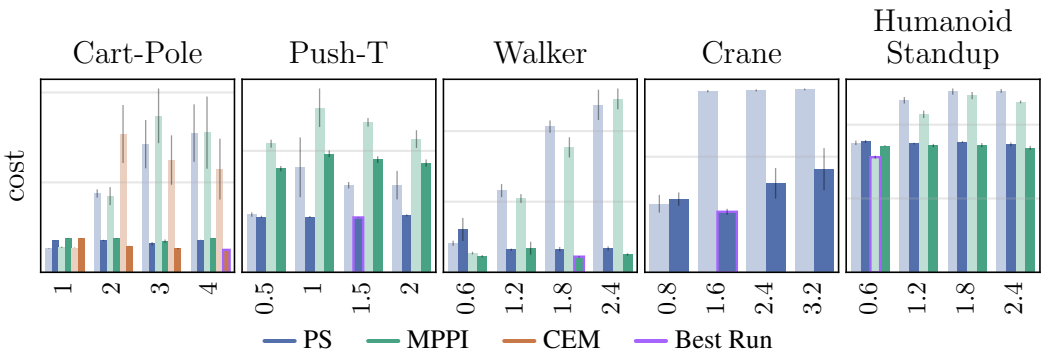


Figure 4: Comparison of empirical cost as a function of time horizon. The darker bars indicate that our horizon splitting algorithm is added to the base sampling-based control method, whereas the lighter bars indicate no splitting. The vertical lines indicate twice the standard error in the mean.

set of bars in figure 4), which indicates that the baselines are well tuned. Generally, we observe that horizon splitting does not always improve over sampling-based methods without modifications for a given task. However, in many tasks, the best configuration has a longer planning horizon, thus illustrating the relevance of splitting.

Table 1: Best Costs and Parameters for Robotics Tasks

	Cart-Pole	Push-T	Walker	Crane	Humanoid
Baselines	4.96 (0.03)	0.46 (0.01)	2.72 (0.21)	**0.30 (0.04)**	**78.34 (0.83)**
With Splitting	**4.81 (0.02)**	**0.45 (0.003)**	**2.21 (0.11)**	**0.26 (0.01)**	85.09 (0.82)
Time Horizon (s)	4	1.5	1.8	1.6	0.6
Samples	1024	128	128	8	1024
Underlying Method	CEM	PS	MPPI	PS	MPPI

Note: Average cost over 5 evaluations with standard deviation in parentheses.

6 Discussion

We introduce sample complexity bounds for sampling-based control methods. Motivated by the theoretical analysis, we formulate a horizon-splitting algorithm that decreases the sample complexity of some sampling-based control methods. We demonstrate the accuracy of our bounds and the horizon splitting on a range of sampling-based methods and control tasks. Our experimental validation reveals non-trivial behavior with increasing time horizons, shows that our bounds are matched by empirical results, and demonstrates the sample efficiency benefit of horizon splitting.

To arrive at our bounds, we required some approximations for which the expected tightness in practice is typically unclear. Importantly, a key step in our derivation involved assuming independence of sampling from timestep to timestep. Indeed, this is standard for current algorithms, but it also opens a clear direction for future work – if we could remove the independence between timesteps, it may be possible to reduce the sample complexity. Additionally, while our model captures the empirical behavior of methods such as MPPI, it does not fully characterize them. MPPI executes a final synthesized mean, rather than a random sample. Extending our bounds to apply to these more nuanced quantities is an interesting direction for future work.

Finally, our analysis motivates a divide-and-conquer algorithm for sampling-based methods and demonstrates the practical applicability for sampling-based control tasks. Nevertheless, there are additional empirical questions that could be investigated. For example, we fixed the sampling to uniform band sampling, because we made no assumption on the source of errors. Yet, it might be possible to assume a distribution on terminal cost errors for a specific task, which can lead to a more efficient resampling procedure. Similarly, Δ is fixed for the whole task, but in many settings it is possible to learn the terminal cost Φ to better approximate the error of Φ to V.

AI USE STATEMENT

We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, implement methods, or interpret results. The rest of the required disclosure tasks are not applicable to this work. We used generative AI tools to create or modify scientific figures or images, to create artifacts, create or edit software code, identify relevant literature, and edit the paper to improve readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have detailed the proofs in the appendix. We build on Hydrax and MuJoCo for our experiments, which are open-source code projects. The experimental parameters for our horizon splitting algorithm are described in Table 2. We will publish our code with the final version of this paper.

ACKNOWLEDGMENTS

Removed for anonymity.

REFERENCES

Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984. doi: 10.1016/S1474-6670(17)61205-9.

Ashwin Gupta and Joseph Moore. Stochastic multiple shooting trajectory optimization via sequential local policy evaluation, 2026. URL https://arxiv.org/abs/2608.03978.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive sampling: Real-time behaviour synthesis with MuJoCo, 2022. URL https://arxiv.org/abs/2212.00541.

Vince Kurtz. Hydrax: Sampling-based model predictive control on GPU with JAX and MuJoCo MJX, 2024. https://github.com/vincekurtz/hydrax.

Vince Kurtz and Joel Burdick. Generative predictive control: Flow matching policies for dynamic and difficult-to-demonstrate task. In *International Conference on Robotics and Automation*, 2026. URL https://arxiv.org/abs/2502.13406.

Chaoyi Pan, Guannan Qu, Guanya Shi, and Zeji Yi. Model-based diffusion for trajectory optimization. In *Advances in Neural Information Processing Systems 37*, pp. 57914–57943, 2024.

Chaoyi Pan, Zeji Yi, John Zhang, Zachary Manchester, Guannan Qu, and Guanya Shi. Hybrid feedback sampling for sample-efficient model predictive control, 2026. URL https://arxiv.org/abs/2608.19443.

Alexandre Piché, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with sequential Monte Carlo methods. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByetGn0cYX.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999. doi: 10.1023/A:1010091220143.

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*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Kevin Tracy, John Z. Zhang, Jon Arrizabalaga, Stefan Schaal, Yuval Tassa, Tom Erez, and Zachary Manchester. The trajectory bundle method: Unifying sequential-convex programming and sampling-based trajectory optimization. 2026. URL https://arxiv.org/abs/2509.26575.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017. doi: 10.2514/1.G001921.

Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, (6):1603–1622, 2018.

Zeji Yi, Chaoyi Pan, Guanqi He, Guannan Qu, and Guanya Shi. CoVO-MPC: Theoretical analysis of sampling-based MPC and optimal covariance design. In *Learning for Dynamics and Control Conference*, volume 242, pp. 1122–1135, 2024. URL https://proceedings.mlr.press/v242/yi24b.html.

Hyung-Jin Yoon, Chuyuan Tao, Hunmin Kim, Naira Hovakimyan, and Petros G. Voulgaris. Sampling complexity of path integral methods for trajectory optimization. In *American Control Conference*, pp. 3482–3487, 2022. doi: 10.23919/ACC53348.2022.9867607.

A Appendix

A.1 Proofs

Proposition 4.6. *Let Assumption 4.4 and Assumption 4.5 hold. Let* $u_{1:T}^{(1)}, \dots, u_{1:T}^{(N)} \sim \mu$. *If*

$$\Pr\Big(\textit{there is some } i \in \{1, \dots, N\} \textit{ s.t. } u_{1:T}^{(i)} \in I_\delta^T(u_{1:T}^\star)\Big) \geq 1 - \varepsilon, \tag{7}$$

then we must have

$$N \geq \left(\frac{1}{p_\delta^T} - 1\right) \log\left(\frac{1}{\varepsilon}\right). \tag{8}$$

Proof. We define two quantities. By Assumption 4.4, there exist $\mu_1, \dots, \mu_T$ such that $\mu = \mu_1 \otimes \cdots \otimes \mu_T$. For the rest of this proof, we omit the dependence on $u_{1:T}^\star$ on the interval I_δ^T for clarity. By Assumption 4.4, the samples are i.i.d., so the probability of sampling in I_δ^T at least once across N samples is given by

$$\begin{aligned} \Pr\Big(\text{There is some } i \text{ such that } u_{1:T}^{(i)} \in I_\delta^T\Big) &= 1 - \Pr\Big(u_{1:T}^{(i)} \notin I_\delta^T \text{ for all } i\Big) \\ &= 1 - (1 - \mu(I_\delta^T))^N. \end{aligned} \tag{20}$$

This probability is at least $1 - \varepsilon$, by assumption, so

$$1 - (1 - \mu(I_\delta^T))^N \geq 1 - \varepsilon \tag{21}$$

$$\implies \quad (1 - \mu(I_\delta^T))^N \leq \varepsilon \tag{22}$$

$$\implies \quad \mu(I_\delta^T) \geq 1 - \varepsilon^{\frac{1}{N}}. \tag{23}$$

Now, by assumption, $p_\delta^T \geq \mu(I_\delta^T)$. Moreover, for $x \in [0,1)$, we have $\log(x) \geq 1 - \frac{1}{x}$, and so $\log(1-x) \geq 1 - \frac{1}{1-x}$. In particular, the previous equation yields

$$p_\delta^T \geq \mu(I_\delta^T) \geq 1 - \varepsilon^{\frac{1}{N}} \tag{24}$$

$$\implies 1 - p_\delta^T \leq \varepsilon^{\frac{1}{N}} \tag{25}$$

$$\implies 1 - \frac{1}{1-p_\delta^T} \leq \log\big(1-p_\delta^T\big) \leq \frac{1}{N}\log(\varepsilon) \tag{26}$$

$$\implies -\frac{p_\delta^T}{1-p_\delta^T} \leq \frac{1}{N}\log(\varepsilon) \tag{27}$$

$$\implies N \geq (1-p_\delta^T)\left(\frac{1}{p_\delta}\right)^T \log\left(\frac{1}{\varepsilon}\right) \tag{28}$$

$$\tag{29}$$

□

Theorem 4.9. *Let Assumptions 4.4, 4.5, and 4.7 hold, and suppose we sample according to an idealized version of algorithm 2 that samples prefix control sequences from the true population conditional distribution. If*

$$\Pr\Big(\textit{there is some } i \in \{1,\dots,N\} \textit{ s.t. } u_{1:T}^{(i)} \in I_\delta^T(u_{1:T}^\star)\Big) \geq 1-\varepsilon, \tag{12}$$

then we must have

$$N \geq p_\delta^{-\frac{T}{K}} \log\left(\frac{1}{\varepsilon}\right) \sum_{k=0}^{K-1} \tilde{p}_\delta^{-k\frac{T}{K}} \left(1 - \tilde{p}_\delta^{k\frac{T}{K}} p_\delta^{\frac{T}{K}}\right), \tag{13}$$

for some $\tilde{p}_\delta \in [p_\delta, 1]$.

Proof. In the language of section 4.1, to correctly sample an action sequence up to time T_{k+1}, we must simultaneously correctly sample from $B_\Delta^{T_k}(x_1)$ and from $\mathcal{U}^{T_k:T_{k+1}}$. These have probabilities

$$\tilde{p}_{k,\delta} = \mu_{1:T_k}(I_\delta^{T_k} | B_\Delta^{T_k}), \quad \text{and} \quad p_\delta^{T_k:T_{k+1}} = \mu_{T_k:T_{k+1}}(I_\delta^{T_k:T_{k+1}}), \tag{30}$$

where we have dropped I_δ's dependence on $u^\star$, to simplify notation.

By Assumption 4.4, the two aforementioned events are independent, so the probability of both is $\tilde{p}_{k,\delta} \cdot p_\delta^{T_k:T_{k+1}}$. As in Proposition 4.6, the required number of samples for the k-th segment is

$$N_k \geq \left(\frac{\tilde{p}_{k,\delta}^{-1}}{p_\delta^{T_k:T_{k+1}}} - 1\right) \log\left(\frac{1}{\varepsilon_k}\right), \tag{31}$$

where we set $\tilde{p}_{0,\delta} = 1$.

By assumption 4.5, there is a p_δ such that

$$p_\delta^{\frac{T}{K}} \geq \mu_{T_k:T_{k+1}}(I_\delta^{T_k:T_{k+1}}) \text{ for all } k = 1,\dots,K.$$

Therefore,

$$N_k \geq \left(\tilde{p}_{k,\delta}^{-1} p_\delta^{-\frac{T}{K}} - 1\right) \log\left(\frac{1}{\varepsilon_k}\right). \tag{32}$$

The first probability is the measure restricted to B_Δ: it assigns 0 probability to events outside B_Δ, and the rest is re-normalized. We can use the same trick to bound $\tilde{p}_{k,\delta}$ from above. However, because we cannot assume factorizability of the conditional distribution, we instead have

$$\tilde{P}_{t,\delta} := \sup_{u_{1:t} \in \mathcal{U}^t} \left[\mu_{1:t}(I_\delta^t(u_{1:t}) | B_\Delta^t(x_1))\right]^{\frac{1}{t}} \tag{33}$$

$$\tilde{p}_\delta := \sup_{t \in \{1,\dots,T\}} \tilde{P}_{t,\delta}. \tag{34}$$

Essentially, $\tilde{p}_\delta$ instead plays the role of a geometric mean, and $\tilde{p}_\delta^{T_k} \geq \tilde{p}_{k,\delta}$ for all k. Moreover, $1 \geq \tilde{p}_\delta \geq p_\delta$, since the conditional measure restricts the set of samples, and so the arguments to the supremum in equation 33 are greater than those in Assumption 4.5.

Of course, $T_k = k\frac{T}{K}$, so

$$N_k \geq (\tilde{p}_\delta^{-k\frac{T}{K}} p_\delta^{-\frac{T}{K}} - 1) \log\left(\frac{1}{\varepsilon_k}\right). \tag{35}$$

Note that the actions up to T_k are only δ-satisfactory if all the actions up to $T_{k'}$ for all $k' \leq k$ are δ-satisfactory. Therefore, ε_k must be at most ε for each k, which gives a lower bound for the number of samples. In conclusion, the total number of samples is the sum

$$\begin{aligned} N = \sum_{k=0}^{K-1} N_k &= \sum_{k=0}^{K-1} \left(\frac{1}{\tilde{p}_\delta^{k\frac{T}{K}} p_\delta^{\frac{T}{K}}} - 1\right) \log\left(\frac{1}{\varepsilon}\right) \\ &= p_\delta^{-\frac{T}{K}} \log\left(\frac{1}{\varepsilon}\right) \sum_{k=0}^{K-1} \left(\frac{1}{\tilde{p}_\delta^{k\frac{T}{K}}} - p_\delta^{\frac{T}{K}}\right) \\ &= p_\delta^{-\frac{T}{K}} \log\left(\frac{1}{\varepsilon}\right) \sum_{k=0}^{K-1} \tilde{p}_\delta^{-k\frac{T}{K}} \left(1 - \tilde{p}_\delta^{k\frac{T}{K}} p_\delta^{\frac{T}{K}}\right). \end{aligned} \tag{36}$$

□

A.2 Experiment Details

LQR Our simulation environment, which runs on the publicly maintained Hydrax repository (Kurtz, 2024), consists of a simulator, along with a control loop. Under the hood, MuJoCo simulates a $0.3\,\mathrm{kg}$ particle subject to external forces using timesteps of $0.02\,\mathrm{s}$. To reduce computational burden, we employ zero-order holds of $0.25s$ for our control actions, and we refer to each hold as a "knot". We plan $3s$ into the future, corresponding to twelve knots, that is $T = 12$, long enough to reach the origin. The planner executes a single simulator step from the synthesized sequence before replanning. As a result, the planner generates 150 plans over the course of each episode, which is also of length $3.0\,\mathrm{s}$.

We cast this simulated system as the linear system

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -10/3 & 0 \\ 0 & 0 & 0 & -10/3 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 10/3 & 0 \\ 0 & 10/3 \end{bmatrix}, \quad Q = I_4, \quad R = I_4. \tag{37}$$

We first cast the problem in continuous time, and then discretize it using standard time-discretization. For example, $A_{\mathrm{disc}} \to e^{A\Delta t}$, where $\Delta t = 0.02\,\mathrm{s}$. At each step, we measure if the full planned trajectory matches (is δ-satisfactory) the open-loop trajectory starting at the current state. Our computed P and R matrices are given by

$$P \approx \begin{bmatrix} 1.6 & 0 & 0.3 & 0 \\ 0 & 1.6 & 0 & 0.3 \\ 0.3 & 0 & 0.19 & 0 \\ 0 & 0.3 & 0 & 0.19 \end{bmatrix}, \quad K \approx \begin{bmatrix} 0.99 & 0 & 0.60 & 0 \\ 0 & 0.99 & 0 & 0.60 \end{bmatrix}. \tag{38}$$

For each episode, the particle starts in a neighbourhood of the origin, and must reach a target at $(0.25\,\mathrm{m}, 0\,\mathrm{m})$. The optimal gain and actions are always within the action limits of $\mathcal{U} = [-1, 1]^2$.

Robotics validation In order to mitigate effects from zero-order hold lengths changing, the zero-order hold is fixed to $0.25s$. Additionally, to avoid effects from the planning horizon being too short for the task, we first determine how long is necessary for a plan to finish the task, the horizon is increased from that point onwards. The total duration of the task is held constant so costs can be compared. The rest of the experiment resembles the LQR greatly. We run 5 episodes for each configuration. Unlike the zero-order hold for knots in LQR, some baselines use cubic splines to interpolate the control sequence. For example, at $\Delta t = 0.02\,\mathrm{s}$ and a knot length of $0.1\,\mathrm{s}$, there are 5 simulator steps for every control knot. The spline interpolates these knots. The temperature refers

to the reweighting coefficient that MPPI uses when updating the mean of its distribution, whereas the noise is the covariance of the distribution. Duration refers to the length of the full episode.

We run each configuration for 5 seeds. The first configuration is picked from the Hydrax examples, since the parameters are tuned for that task and sampling algorithm. From there, we keep all parameters constant, except we increase the planning horizon. With horizon splitting, we use $K = 2, 4, 6, 8$ to keep the number of knots per sub-horizon constant for all tasks.

More information can be found at the Hydrax repository. Briefly, the Cart-Pole task is a standard swing-up task to stabilize an inverted pendulum. Push-T involves pushing a "T" into a target position. Walker has a 2 limb humanoid walking as far as possible in a given direction. Crane involves extending and retracting a string on a rod for its massive end to reach a target position. Humanoid standup involves a G1 humanoid standing up from a random position on the ground.

Table 2: Hyperparameters for Horizon Splitting Ablation

	Samples N	Temperature	Noise	Knot Length (s)	Duration	Spline
Cart Pole	128	0.1	0.3	0.25	6.0	cubic
Push-T	128	0.1	0.4	0.1	10.0	zero
Walker	128	0.1	0.5	0.15	5.0	zero
Crane	8	N/A	0.5	0.4	5.0	zero
Humanoid	128	0.1	0.3	0.2	6.0	zero