Optimal Noise Control of Sampling-Based Predictive Control for Inference Time Scaling

Benjamin Riviere and Joel Burdick. Paper-ID [41]

***Abstract*—Robots that are controlled by evaluating an offline trained model can fail in the presence of out-of-distribution data. Real-time optimization, also known as inference time scaling, can be a complementary approach to make learning-based systems robust to this error. One such method, sampling-based predictive control (SBPC), uses finite samples to approximate gradient steps in trajectory optimization. SBPC methods are gaining attention because they are less sensitive to dynamical modeling errors, allow for naturally parallelizable implementations, and they can be used in combination with pretrained models for inference time scaling and out-of-domain generalization. However, the performance of SBPC is dependent on hyperparameters, the most important of which is the noise model that controls the finite sample generation process. Whereas existing work considers the noise model as a problem-specific hyperparameter, we take the perspective of noise as a control input to the SBPC planning process, develop the corresponding optimal control problem, and propose a hierarchical search-and-sample solution. In particular, we parameterize the noise model with parameters that control the overall noise level and the noise level as a function of simulation time, and optimize these parameters over the solver iterations with Monte Carlo Tree Search. Our approach removes hyperparameters, maintains finite-time theoretical convergence guarantees, and improves empirical performance on robot planning problems.**

I. Introduction

Machine learning models are rapidly being adopted for robot planning and control. The standard strategy is to generate a dataset, train a model offline, freeze the weights, and deploy the model online. Perhaps the most fundamental challenge of this approach is out-of-domain distribution data – when the data distribution during the robot's deployment is different from the distribution on which the model was trained. A possible solution is to augment the model evaluation with additional inference-time computation to address the problem of new state distributions or dynamics.

Sampling-Based Predictive Control (SBPC) is promising approach at the intersection of existing ideas in robot learning and optimal control. SBPC with learned models can be interpreted as a diffusion process from the model's prior distribution to a target optimal distribution. SBPC can also be interpreted through the lens of classical optimization as an approximate smoothed gradient flow.

However, the performance of SBPC is highly dependent on the choice of parameters such as time horizon, time discretization, number of finite samples, and noise schedule. Whereas prior work treats these as manually-tuned problem-specific hyperparameters, our work treats these as control inputs to the SBPC gradient flow dynamical system, and optimizes them for system performance, as shown in Fig. 1.

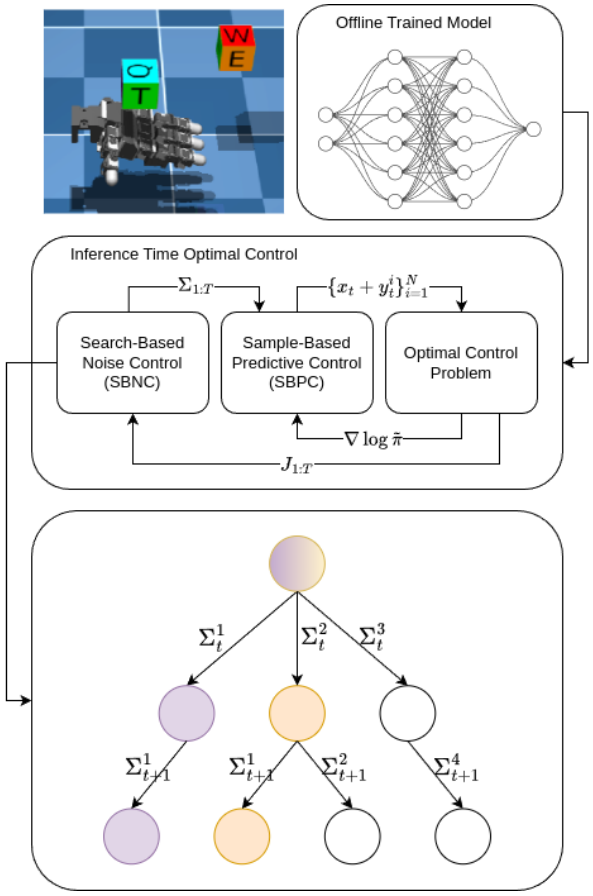


Fig. 1: We propose a hierarchical formulation of inference-time optimal control as search-based noise control and sampling-based predictive control. Prior work uses either constant noise (shown in purple) or a fixed annealing schedule (shown in yellow), whereas we propose an adaptive optimization. Here Σ_t^i are covariance matrices, $x_t + y_t^i$ are perturbed control sequences, J is the cost-to-go, and $\tilde{\pi}$ is the smoothed target distribution.

In this preliminary work, we isolate the effect of the noise schedule on SBPC performance, and introduce new methods to adaptively schedule the noise in the solution process.

Our work makes the following contributions:

- We propose a novel and mathematically unified hierarchical formulation of the inference-time optimal control problem using SBPC as a mid-level component.
- We propose a search and sample algorithm using noise model parameterization and Monte Carlo Tree Search that

solves the hierarchical problem, removes hyperparameters, and maintains theoretical guarantees.
- We validate our approach on numerical experiments, where our solution improves performance and robustness to hyperparameters.

II. Related Work

Relevant to our work is sampling-based predictive control (SBPC) methods in robotics that approximate gradients with finite samples. These methods are mature and include Cross-Entropy Method (CEM) [1] and Model Predictive Path Integral (MPPI) control [2]. Because SBPC is easily parallelized, recent trends in GPU computation have boosted their performance, enabling new results in real-time manipulation [3] and locomotion [4], as well as competitive baselines [5]. Furthermore, it is well known that SBPC can be integrated with learned models [6] to benefit from prior knowledge while remaining robust to out-of-domain distributions. SBPC with learned and uncertain dynamics is a rich area that spans economic cost functions [7], active sensing [8], partial information [9] and robustness to uncertainty [10].

SBPC can be interpreted as a dynamical system, and its convergence is perhaps best theoretically understood through its stochastic extension, the Annealed Langevin Dynamics (ALD) [11]. Recently, it has been shown that the ALD have better convergence properties than standard Langevin Dynamics [12–14], where the time varying noise schedule enables the ALD to avoid local minima through an annealing-like procedure. Although it is known that the noise schedule controls the exploration-exploitation trade-off, the selection of a noise schedule to optimize convergence rate and real-time performance remains an open question.

Similar to our approach, there exists work on optimal noise scheduling. For example, in the diffusion and score-based model literature, a standard treatment is to pre-specify a linear [15], trigonometric [16] or polynomial [17] rate for all problems. Ideally, however, we would like a problem-specific adaptive solution to maximize convergence rate, e.g. if the problem is unimodal and smooth, the noise should degrade very quickly. Recent work considers adaptive timesteps and noise levels such as using a fine-resolution teacher to inform a coarse-resolution student to do fast model passes [4] or learning adaptive noise levels [18], but these methods require pretraining and data from the target distribution.

It is possible to adapt noise levels online, such as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [19]. While CMA-ES updates noise based on feedback from the current sample evaluations, our approach updates noise using information from future sample evaluations using the tree search's backup operation. To the best of our knowledge, we are the first to explore adaptive noise processes using predictive information feedback. We believe this is a promising direction because the noise process must balance exploration and exploitation: high noise improves coverage and low noise improves local optimality, and therefore it is difficult to optimize noise levels using only local information.

III. Background and Problem Statement

We start with a standard formulation of the optimal control problem: let $S \subseteq \mathbb{R}^n$ be the robot state space, $U \subseteq \mathbb{R}^m$ be the action space, $f : S \times U \to S$ be the system dynamics, $c : S \times U \to \mathbb{R}_{>0}$ be the stage cost, $d : S \to \mathbb{R}_{>0}$ be the terminal cost, and $K \in \mathbb{Z}_{>0}$ be the plan horizon. Let $J : S \times U^K \to \mathbb{R}_{>0}$ be the cost-to-go of an action from an initial state: $J(s_0, u_{1:K}) = \sum_{k=1}^{K} c(s_k, u_k) + d(s_K)$ where k is the timestep index and the state evolves according to the dynamics $s_{k+1} = f(s_k, u_{k+1})$, $\forall k \in [1, \dots, K]$. The optimal control problem is to find the control sequence that minimizes cost-to-go:

$$u^*_{1:K} = \arg\min_{u_{1:K}} J(s_0, u_{1:K}) \tag{1}$$

The optimal control problem can be solved with sampling-based predictive control (SBPC) [2, 20–22]. Fixing the initial robot state s_0, SBPC iterations can be interpreted as a dynamical system that evolves the control sequence with smoothed gradient flow:

$$x_{t+1} = x_t - \eta_t \nabla \log \tilde{\pi}(x_t, \Sigma_t) \tag{2}$$

where, x_t is the control sequence $x_t = u^t_{1:K}$, x_0 is the initial control sequence sampled from a prior distribution $x_0 \sim \pi_{\text{prior}}$, η_t is the stepsize, $\tilde{\pi}$ is the smoothed target distribution $\tilde{\pi}(x, \Sigma) = (\pi \star \phi_{0,\Sigma})(x)$, π is the target distribution $\pi(x) \propto e^{J(s_0,x)}$, $\phi_{\mu,\Sigma}$ is a normal distribution with mean μ and covariance Σ, and t is an SBPC iteration index. The smoothed gradient, sometimes called the score, is approximated with finite samples:

$$\nabla \log \tilde{\pi}(x, \Sigma) \approx -\Sigma^{-1} \sum_{i=1}^{N} w_i y_i \tag{3}$$

where $y_i \sim \phi_{0,\Sigma}$, $w_i = e^{J(x+y_i)} / \sum_{j=1}^{N} e^{J(x+y_j)}$, and N is the number of samples. This identify is shown in [20, 22].

The interpretation of SBPC as smoothed gradient flow (or its noisy extension, the Annealed Langevin Dynamics) is useful for our algorithm design and for application of existing theoretical convergence analyses [12, 23]. In the limit $\Sigma \to 0$, $\tilde{\pi} \to \pi$, and the iterations reduce to: $x_{t+1} = x_t - \eta \nabla J(s_0, x_t)$, recovering pure gradient flow.

The current algorithm iterate x_t is a function of the initial condition x_0 and the past noise schedule $\Sigma_{1:t}$, i.e. $x_T = x_T(x_0, \Sigma_{1:T})$. Whereas existing work in SBPC treats the noise schedule as a problem-specific hyperparameter, we introduce the optimal noise control problem, which is to find the *noise schedule* that optimizes SBPC performance:

$$\Sigma^*_{1:T} = \arg\min_{\Sigma_{1:T}} J(s_0, x_T(x_0, \Sigma_{1:T})) \tag{4}$$

where T is the number of SBPC iterations. This optimization framework provides a unifying framework for existing SBPC. See Fig. 1.

The optimization problems (1) and (4) are similar because they are a hierarchical formulation of the optimal control problem, Fig. 1. Whereas conventional hierarchies use manually

designed abstraction layers (e.g. geometric path planning then tracking control), ours uses the same objective and decision-variable space (augmented with noise Σ), providing a simple and mathematically unified hierarchy with a minimal number of hyperparameters.

IV. Method

Our approach to the optimal noise control problem (4) is to parameterize the noise schedule, formulate a discrete Markov Decision Process, (MDP) [24, 25] and solve it with Monte Carlo Tree Search (MCTS) [26].

The key design of our algorithm is in the parameterization of the noise:

$$\Sigma(\theta, \psi) = \theta \left(\text{diag} \left(\frac{1}{K}, \frac{2}{K}, \ldots, 1 \right)^{\psi} \right) \otimes I_m \tag{5}$$

where $\otimes$ is the Kronecker product. Our parameterization reduces search space complexity from $\Sigma_{1:T} \in \mathbb{S}^{mKT \times mKT}$ to $\theta_{1:T} \in \mathbb{R}^{2T}$.

Next, we discuss the effect of the parameters θ and ψ in (5), also shown in Fig. 2. The first parameter, θ, controls the overall noise scale. Based on annealing literature, we expect that the optimal evolution of this parameter is to start large or even grow during a transient phase, but eventually decay over solver iterations (denoted by t).

The second parameter, ψ controls the noise of the control actions based on their simulation timestep (denoted by k). The vector $\left(\frac{1}{K}, \frac{2}{K}, \ldots, 1\right)$ is a straight line from $(0, 0)$ to $(K, 1)$, and raising it to a power controls the warping – a larger value of ψ causes the noise scale to grow more slowly, delaying the admittance of noise into the trajectory as a function of simulation time. We expect the optimal evolution of this parameter is to start small and increase over iterations, corresponding to decreasing the exploration noise in the actions at early timestep in the trajectory and "committing" to actions. In this context, we can interpret this parameter as controlling the interpolation between model-predictive control (MPC) and open-loop-planning. We present experimental results consistent with this discussion in Sec. V.

We define the search MDP as follows: the state is the iterate and noise level parameterization $z_t = [x_t, \theta_t] \in Z$, the action $a_t \in A$ scales the noise parameters where A is a discrete set, the cost $\tilde{J} : Z \to \mathbb{R}$ is the cost-to-go, and the horizon is the SBPC budget T. The algorithm state transition, $F : Z \times A \to Z$ is:

$$x_{t+1} = x_t - \eta_t \nabla \log \tilde{\pi}(x_t, \Sigma_t(\theta_t, \psi_t)) \tag{6}$$

$$\theta_{t+1} = \theta_t a_{1,t} \tag{7}$$

$$\psi_{t+1} = \psi_t a_{2,t} \tag{8}$$

The MDP can be written compactly as $\langle Z, A, F, \tilde{J}, T \rangle$.

We solve this MDP with Monte Carlo Tree Search (MCTS). Nodes of the tree store algorithm states z, edges store actions a, and edges evaluate the MDP transition function F and reward function R. The tree's policy selects which actions to take based on a balance of exploration and exploitation

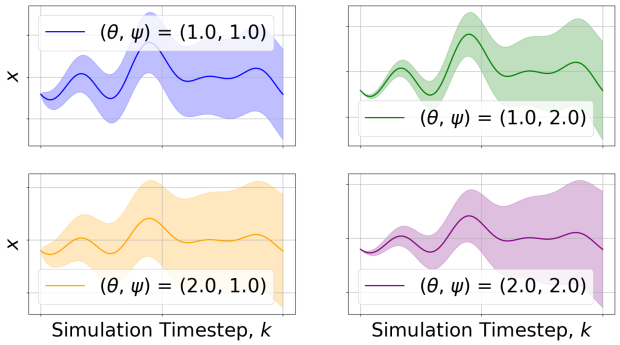


Fig. 2: We show a single solution iterate x_t and the first quartile of the distribution of perturbed signals $x_t + y_t^i$ for different parameters θ and ψ. The θ parameter controls the overall noise level, and the ψ parameter controls the time windowing effect.

using the upper confidence bound selection rule. Because we use a standard implementation, we refer the reader to the existing literature [26, 27]. The pseudocode implementation of our algorithm is detailed in Alg. 1.

MCTS is well-suited for this problem because the noise process must balance exploration and exploitation: high noise improves coverage and low noise improves local optimality. Therefore, it is difficult to optimize noise levels using only local information, and predictive information feedback through the tree's backup operator is necessary to adapt the noise process.

Under mild assumptions, our algorithm has provable convergence. Let $\psi^*_{1:T}$, $\theta^*_{1:T}$ be the best noise parameterization:

$$\theta^*_{1:T}, \psi^*_{1:T} = \underset{\theta_{1:T}, \psi_{1:T}}{\arg\min}\, J(s_0, x_T(x_0, \Sigma_{1:T}(\theta_{1:T}, \psi_{1:T})) \tag{9}$$

and let $\tilde{J}^*(s_0)$ be the corresponding cost-to-go, $\tilde{J}^*(s_0) = J(s_0, x_T(x_0, \Sigma_{1:T}(\theta^*_{1:T}, \psi^*_{1:T}))$. Because the MDP is discrete, our algorithm inherits the value convergence of MCTS [26] to $\tilde{J}^*(s_0)$, and we can apply triangle inequality with the cross term $\tilde{J}^*(s_0)$ to say our search converges towards the solution of the original optimal control problem (1):

$$|J^*(s_0) - \mathbb{E}[\hat{J}_l(s_0)]| \leq \frac{c_0}{\sqrt{l}} + \epsilon \tag{10}$$

where $\mathbb{E}[\hat{J}_l(s_0)]$ is the MCTS root node cost-to-go estimate, l is the number of iterations of the MCTS algorithm, $c_0 > 0$ is a problem-specific constant, and ϵ is the hypothesis space error $\epsilon = |\tilde{J}^*(s_0) - J^*(s_0)|$.

In other words, if there exists a sequence of parameters $\theta_{1:T}$ and $\psi_{1:T}$ such that the noise process drives the SBPC dynamics to the optimal cost-to-go, then our algorithm will have good asymptotic performance.

V. Experiments

We present two numerical experiments: first, we isolate the inference-time optimal control problem and show that our optimal noise control method improves convergence and

Algorithm 1: Search-Based Noise Control (SBNC)

Input: Initial robot state s_0, Initial algorithm state $z_0 = [x_0, \theta_0, \psi_0]$
Output: Cost-to-go Estimate $\hat{J}$
$\hat{J} = J(s_0, x(z_0))$;
while *computation budget remains* **do**
  $z \leftarrow z_0$;
  // Selection
  while z *is expanded* **do**
    $z \leftarrow \max_{z' \in \text{children}(z)} \frac{\text{value}(z')}{\text{visits}(z')} + \frac{\sqrt{\text{visits}(z)}}{\text{visits}(z')}$;
  // Expansion
  $z \leftarrow F(\text{parent}(z), \text{action\_to\_node}(z))$;
  // Save Best
  $\hat{J} \leftarrow \min(\hat{J}, J(s_0, x(z)))$;
  // Simulation
  $v \leftarrow \text{Rollout}(z)$;
  // Backpropagation
  while z *is not null* **do**
    $\text{visits}(z) \leftarrow \text{visits}(z) + 1$;
    $\text{value}(z) \leftarrow \text{value}(z) + v$;
    $z \leftarrow$ parent of z;
return $\hat{J}$

robustness of sampling-based planners. Second, we show that combining learned models and sampling-based planning can improve convergence rate and compensate for sim-to-real error. These two experiments show that our method improves SBPC convergence, and this improvement immediately transfers to fundamental problems in data-driven planning.

For both experiments, we use the following implementation details. For task and baseline implementations, we use the Hydrax repository [28] built on the vectorization and parallelization tools in JAX [29] and MuJoCo MJX [30]. Whereas the Hydrax repository runs SBPC in a model-predictive-control mode (i.e. plan, take first action, replan), we focus on only the planning subroutine. Unless otherwise specified, we use the following parameters: for our method, we initialize noise parameters as $\theta_0 = 0.3$ and $\psi_0 = 1$ and we use $A = \{0.2, 1.0, 5.0\}^2$ for all tasks. For baseline methods, we use the existing tuned parameters in Hydrax for all tasks. Each algorithm iteration corresponds to one step of (2) and computes the systems dynamics and cost functions NK times, where N is the number of finite samples used to estimate the gradient in (2)) and K is the number of timesteps in one trajectory. We give every method 100 total solver iterations. Because the computation of each solver iteration dominates the runtime and each solver uses the same number of iterations, the runtime between all of the solvers is approximately equivalent.

A. Experiment 1: Inference Time Compute

We evaluate on the CubeRotation task, where the LEAP [31] hand is tasked to manipulate the cube to a goal orientation, shown in Fig. 1. This problem has 23 degrees-of-freedom, and is considered a challenging problem because of rich contact dynamics with multiple switching modes and high dimensionality. We implement our method, SBNC, and compare performance with existing SBPC algorithms: MPPI [2], MBD [20], DIAL [21], and Predictive Sampling [5].

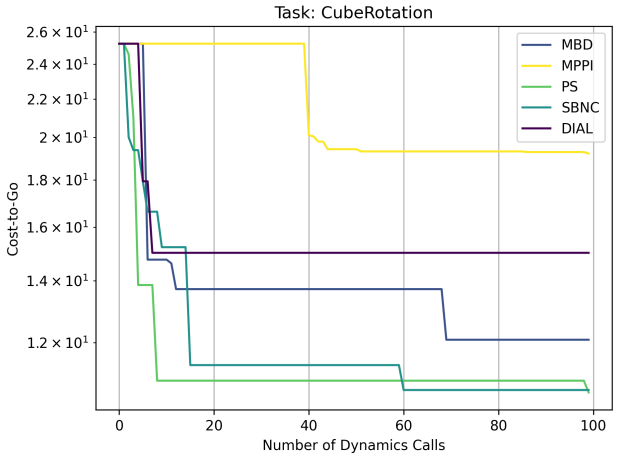


Fig. 3: Performance plot for CubeRotation with nominal noise levels.

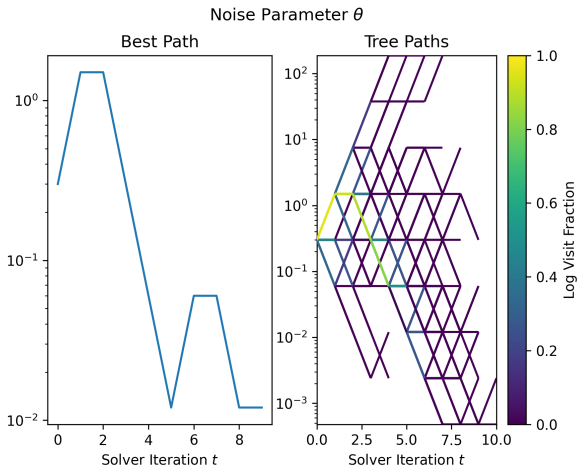
First, we evaluate the performance of different solvers with the nominal parameter settings, i.e. $\theta_0 = 0.3$, and we present the results in Fig. 3. At this setting, the solvers all have similar performance, with a slight advantage to our solver. Roughly speaking, a cost-to-go on the order of 10 corresponds to a terminal state where the cube is rotated 90 degrees towards the goal, and a cost-to-go on the order of 5 corresponds to a second 90 degrees rotation and completing the task.

Next, we run the same experiment, except we scale and shrink the initial noise level $\theta_0 = 0.30$ by a factor of 2 two times, and present the results in Tab. I. In this case, the performance of other methods degrades sharply, whereas our approach is robust to suboptimal hyperparameter initialization and maintains a low average cost-to-go. This behavior is expected because our planner, unlike other approaches, adapts the noise level θ to grow or shrink over solver iterations to optimize performance.

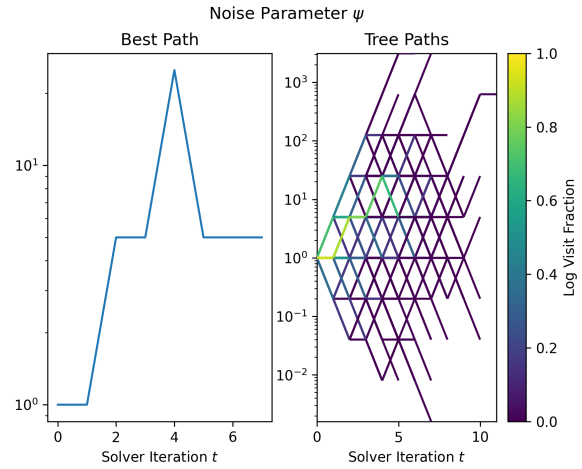
TABLE I: Cost-to-Go after 100 Calls with Varying θ_0.

θ_0	DIAL	MBD	MPPI	SBNC (Ours)
0.075	25.25	24.27	12.27	12.74
0.15	15.38	16.10	19.54	4.39
0.30	15.00	12.08	19.21	10.36
0.60	17.73	18.36	16.23	10.10
1.20	5.42	13.41	10.52	12.45
Average	15.76	16.84	15.55	10.01

We can analyze the noise process discovered by our solver by visualizing the θ and ψ search traces, shown in Fig. 4a/b, where we show the best path on the left and the tree paths on the right. The best θ path grows in magnitude in early solver iterations and then shrinks, matching the expected annealing



(a) Solver iteration vs θ noise parameter.



(b) Solver iteration vs ψ noise parameter.

Fig. 4: The path with the highest reward is shown in the left, and the tree of all paths, colored by their relative visit count, is shown in the right. The relative visit count of a node is an indicator of its value because the exploration and exploitation balance of Monte Carlo Tree Search.

pattern. Discovering, rather than designing, the path of θ is important for flexibility and performance – if the original θ_0 is too small, our solution can first allow the noise level to grow to escape some local minima before the shrinking the noise to do local refinement. We believe local minima is the most likely explanation for poor performance of monotonically decreasing annealing solutions when initialized small noise levels, as shown in Tab. I. The best ψ path grows with solver iterations, and this behavior corresponds to shrinking the exploration noise of the control action at the later timesteps of the solution. Automatically discovering optimal ψ path can be interpreted as deciding when to commit to an action, and roughly bridges open loop planning and model predictive control behaviors.

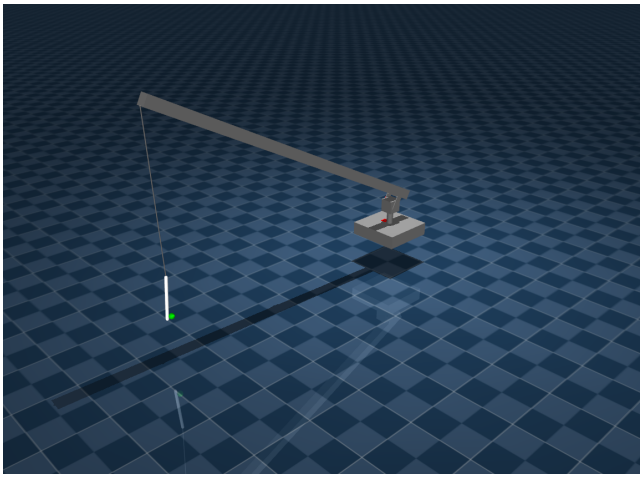


Fig. 5: Crane System: the payload is the gray cylinder, the target is the green dot, and we control the slew and luff at the crane base, as well as the hoist at the tip of the boom.

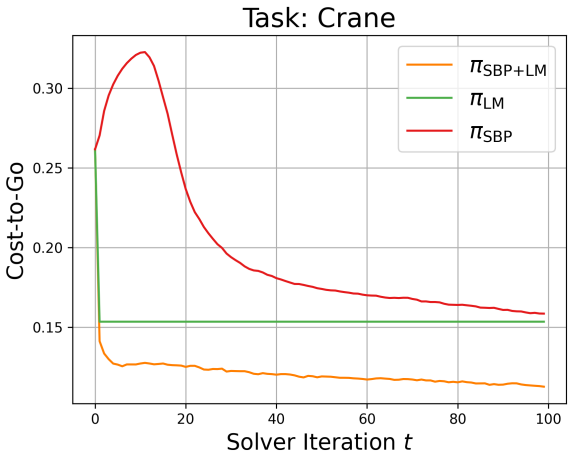


Fig. 6: Solver iteration vs Cost-to-Go for the Crane task, where π_{SBP} is the sampling-based predictive control, $\pi_{\text{SBP+LM}}$ is the sampling-based predictive control with a learned model, and π_{LM} is just the learned model.

B. Experiment 2: Sim to Real Adaptation

We evaluate on the Crane task in Hydrax, where the crane is tasked to move a payload to a target location, shown in Fig. 5. This problem has a 17 dimensional state space and its dynamics are underactuated and sensitive to actuator gains and mass parameters. We construct a hybrid planner that combines SBPC with a learned model, and we empirically demonstrate this hybrid planner has better convergence rate than SBPC alone, and better robustness to the sim-to-real problem than the learned model alone.

We consider three policies: let π_{SBPC} be any SBPC from V-A, π_{LM} be a learned model, and let $\pi_{\text{SBPC+LM}}$ be the hybrid planner. The hybrid policy $\pi_{\text{SBPC+LM}}$ is the same as the SBPC, but warm-started from the learned model (i.e. the learned model is the prior distribution in (2)). The learned

model is trained via standard self-play loop: initialize the learned model as a zero mapping, generate a dataset of robot state and action sequences using the hybrid model, update the learned model parameters with supervised learning, and then iterate over the last two steps.

$$\pi^0_{\text{LM}} : S \to 0_{m \times K} \tag{11}$$

$$\mathcal{D}^l = \{(s^i_0, x^i) \mid s^i_0 \sim \rho, x^i \sim \pi^{l-1}_{\text{SBPC+LM}}(s^i_0)\}^{N_{\mathcal{D}}}_{i=1} \tag{12}$$

$$\pi^l_{\text{LM}} = \arg\min_{\pi} \sum_{(s^i_0, x^i) \in \mathcal{D}^l} \|\pi(s^i_0) - x^i\|_2 \tag{13}$$

where $l \in [1, N_l]$ is the learning iteration.

We consider two dynamics models: let f_{sim} be the simulation dynamics model and let f_{real} be the real dynamics model. To construct f_{real}, we modify the nominal f_{sim} using the default domain randomization parameters in Hydrax, which include variations in mass, friction coefficient, and actuator gains. To model the sim-to-real effect, we train the learned model using data generated from f_{sim} and evaluate the policies using f_{real}. We assume that all of the SBPC policies evaluate their gradient updates (2) with the real dynamics f_{real}, which is a reasonable assumption for systems that have runtime system identification or adaptive control components [32].

The solvers are evaluated and the results are presented in Fig. 6. As desired, the hybrid policy has fewer iterations-to-convergence than the SBPC policy, and the hybrid policy has a smaller terminal cost-to-go compared to the learned policy, demonstrating robustness to the sim-to-real gap. This example demonstrates that SBPC can be used to repair error from out-of-distribution data.

VI. Conclusion

We study the inference-time optimal control problem as a hierarchical combination, and presented a search-based noise control and sampling-based predictive control solution. We show that adapting noise parameters is practical approach for ensuring robust performance, and is a mechanism to discover intelligent meta-behavior like committing to early actions during planning in order to reduce search complexity. We also show how efficient inference-time optimal control can directly contribute to out-of-domain generalization problems for learned models. In future work, we will extend this framework to control other parameters in the SBPC process, and we will further develop simulation and hardware experiments.

References

[1] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999.

[2] Bogdan Vlahov, Jason Gibson, Manan Gandhi, and Evangelos A Theodorou. Mppi-generic: A cuda library for stochastic optimization. *arXiv preprint arXiv:2409.07563*, 2024.

[3] Albert H Li, Preston Culbertson, Vince Kurtz, and Aaron D Ames. Drop: Dexterous reorientation via online planning. *arXiv preprint arXiv:2409.14562*, 2024.

[4] Jianning Pei, Han Hu, and Shuyang Gu. Optimal stepsize for diffusion sampling, 2025. URL https://arxiv.org/abs/2503.21774.

[5] 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.

[6] Thomas Power and Dmitry Berenson. Variational inference mpc using normalizing flows and out-of-distribution projection. *arXiv preprint arXiv:2205.04667*, 2022.

[7] Sergio Lucia, Joel AE Andersson, Heiko Brandt, Moritz Diehl, and Sebastian Engell. Handling uncertainty in economic nonlinear model predictive control: A comparative case study. *Journal of Process Control*, 24(8):1247–1259, 2014.

[8] Elena Arcari, Lukas Hewing, Max Schlichting, and Melanie Zeilinger. Dual stochastic mpc for systems with parametric and structural uncertainty. In *Learning for Dynamics and Control*, pages 894–903. PMLR, 2020.

[9] Robert Dyro, James Harrison, Apoorva Sharma, and Marco Pavone. Particle mpc for uncertain and learning-based control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7127–7134. IEEE, 2021.

[10] S Thangavel, S Lucia, R Paulen, and S Engell. Dual robust nonlinear model predictive control: A multi-stage approach. *Journal of process control*, 72:39–51, 2018.

[11] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[12] Paula Cordero-Encinar, O. Deniz Akyildiz, and Andrew B. Duncan. Non-asymptotic analysis of diffusion annealed langevin monte carlo for generative modelling, 2025. URL https://arxiv.org/abs/2502.09306.

[13] Wei Guo, Molei Tao, and Yongxin Chen. Provable benefit of annealed langevin monte carlo for non-log-concave sampling. *arXiv preprint arXiv:2407.16936*, 2024.

[14] Xuedong He, Xiaolu Tan, and Ruocheng Wu. Convergence of simulated annealing using kinetic langevin dynamics. *Electronic Journal of Probability*, 29:1–31, 2024.

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

[16] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[17] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[18] Subham Sekhar Sahoo, Aaron Gokaslan, Chris De, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and

C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 105730–105779. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/bee43378b65ec195a67f24709469dcaf-Paper-Conference.pdf.

[19] Raymond Ros and Nikolaus Hansen. A simple modification in cma-es achieving linear time and space complexity. In *International conference on parallel problem solving from nature*, pages 296–305. Springer, 2008.

[20] Chaoyi Pan, Zeji Yi, Guanya Shi, and Guannan Qu. Model-based diffusion for trajectory optimization, 2024. URL https://arxiv.org/abs/2407.01573.

[21] Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. *arXiv preprint arXiv:2409.15610*, 2024.

[22] Vince Kurtz and Joel W Burdick. Generative predictive control: Flow matching policies for dynamic and difficult-to-demonstrate tasks. *arXiv preprint arXiv:2502.13406*, 2025.

[23] Arnak S Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(3):651–676, 2017.

[24] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[25] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. 1*. Athena Scientific, 4th edition, 2012.

[26] Benjamin Rivière, John Lathrop, and Soon-Jo Chung. Monte carlo tree search with spectral expansion for planning with dynamical systems. *Science Robotics*, 9 (97):eado1010, 2024. doi: 10.1126/scirobotics.ado1010.

[27] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

[28] Vince Kurtz. Hydrax: Sampling-based model predictive control on gpu with jax and mujoco mjx, 2024. https://github.com/vincekurtz/hydrax.

[29] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

[30] M.X. Authors. Mujoco xla (mjx). https://mujoco.readthedocs.io/en/stable/mjx.html, 2025.

[31] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023.

[32] Elena Sorina Lupu, Fengze Xie, James A Preiss, Jedidiah Alindogan, Matthew Anderson, and Soon-Jo Chung. Magic vfm-meta-learning adaptation for ground interaction control with visual foundation models. *IEEE Transactions on Robotics*, 2024.