A Study of Action-Sequence Diffusion for Open-Loop Control under Dry Friction and Stiction

Eric Aislan Antonelo[1]

[1] Department of Automation and Systems Engineering (EAS)
Federal University of Santa Catarina (UFSC)
Florianópolis, SC, Brazil

eric.antonelo@ufsc.br

Abstract. *Diffusion models have recently emerged as expressive generative priors for planning and control. This paper studies Action Diffusion, an action-sequence diffusion formulation used as an open-loop proposal distribution for a point-mass system with dry friction and stiction. In this benchmark, motion starts only when the applied input exceeds a static-friction threshold, so effective controls occupy a small and temporally structured subset of the action-sequence space. A compact conditional 1D U-Net generates bounded control sequences conditioned on initial and target states. We compare it with uniform random shooting, random shooting from the same structured dataset prior, and the Cross-Entropy Method (CEM). Results show that Action Diffusion reduces terminal error and stuck steps, especially in low-sample regimes. These results indicate that conditional diffusion provides an effective mechanism for generating temporally coherent control sequences that overcome stiction by conditioning and recombining structured control primitives from the training prior for state-to-state open-loop control.*

1. Introduction

Recent advances in generative modeling have opened new perspectives for sequential decision-making and control. In particular, diffusion models [Ho et al. 2020, Song et al. 2021] have been adapted to generate trajectories or action sequences conditioned on goals, rewards, or observations [Janner et al. 2022, Ajay et al. 2023, Chi et al. 2023]. While the Brazilian artificial intelligence community has recently explored diffusion models primarily for generative tasks involving static data and image processing [Kemmer et al. 2023], their application as expressive priors for continuous physical control and action-sequencing remains largely underexplored. To bridge this gap, this paper studies an Action Diffusion formulation for control problems.

Most empirical studies of diffusion-based control focus on high-dimensional settings, such as robotic manipulation or offline reinforcement-learning benchmarks [Janner et al. 2022, Ajay et al. 2023, Chi et al. 2023, Levine et al. 2020]. These environments are important, but their complexity often makes it difficult to inspect the generated control signals and to identify why a generative prior helps. A complementary approach is to study simpler systems in which the relevant physical structure is explicit.

This paper follows that direction and considers a one-dimensional point-mass system with dry friction and stiction. Stiction creates a dead zone: when the system is nearly

at rest, inputs below the static-friction threshold do not initiate motion. From a sampling perspective, this is challenging because effective control sequences must have the correct sign, magnitude, and timing. Uniformly sampled sequences may exceed the threshold occasionally, but often do so in the wrong direction or for an inappropriate duration. Dataset-prior sampling can include useful high-amplitude segments, but without conditioning it may still generate sequences suited to a different target displacement.

We study this problem through an action-sequence diffusion formulation for state-to-state open-loop control. Instead of denoising full state-action trajectories as in [Janner et al. 2022], the model represents a conditional distribution over control sequences, $p_\theta(u_{0:H-1} \mid x_0, x^\star)$, where x_0 is the initial state and $x^\star$ is the desired terminal state. Candidate control sequences sampled from this distribution are rolled out through the known dry-friction dynamics and ranked by terminal error. This separates the generative modeling of useful control patterns from the physical evaluation of the resulting trajectories, and makes the diffusion model a learned proposal distribution for sampling-based open-loop planning. We refer to this formulation as *Action Diffusion*, since diffusion is applied directly to the action/control sequence rather than to the full state-action trajectory.

This paper asks whether conditioning a diffusion model on the initial and target states provides a more sample-efficient proposal distribution for non-smooth open-loop control when the training prior is structured but unconditioned. Our hypothesis is that conditioning this prior on $(x_0, x^\star)$ increases the probability of sampling sequences with suitable sign, magnitude, and timing. Since this prior already contains "kick", medium-amplitude, and near-zero segments, the model is not claimed to discover these primitives from scratch.

The main contributions of this work are:

- A compact action-sequence diffusion formulation for state-to-state open-loop control, in which the model learns $p_\theta(u_{0:H-1} \mid x_0, x^\star)$ and is used as a conditional proposal distribution whose samples are evaluated by rollout through the known dry-friction dynamics.
- A comparison against uniform random shooting, dataset-prior random shooting, and CEM, highlighting the difference between an unconditioned structured prior and a learned conditional prior.
- An analysis of the generated controls showing that the model conditions and recombines structured control segments into interpretable threshold-crossing, motion, and settling phases.

2. Related Work

Diffusion models have recently been used as generative priors for planning and control. Diffuser [Janner et al. 2022] denoises full state-action trajectories and uses sampling to select useful behaviors, while Decision Diffuser [Ajay et al. 2023] formulates decision making as conditional generative modeling with conditioning variables such as returns or constraints. Diffusion Policy [Chi et al. 2023] generates action sequences for visuomotor robotic manipulation, and diffusion-based behavior cloning has also been investigated for autonomous driving in simulated environments [Machado and Antonelo 2025]. Our study is complementary to these works: instead of targeting high-dimensional benchmarks, we

focus on a low-dimensional non-smooth system where the temporal structure of the generated controls can be inspected directly. The main modeling distinction is that Action Diffusion denoises only the control sequence and uses the known dynamics externally for rollout and ranking, rather than generating full state-action trajectories.

The proposed evaluation is also related to sampling-based control. Random shooting generates candidate control sequences and selects the best after rollout, while adaptive methods such as CEM [Rubinstein 1999] iteratively refine the proposal distribution. MPPI-style methods [Williams et al. 2017] are widely used because they can handle nonconvex objectives and do not require differentiability. In systems with stiction, however, large regions of the action space are ineffective because inputs below the static-friction threshold do not move the system. This makes the choice of proposal distribution central: uniform sampling, dataset-prior sampling, CEM, and conditional diffusion differ mainly in how they allocate samples in the control-sequence space.

Finally, friction is a classical source of nonlinearity and non-smooth behavior in mechanical systems [Armstrong-Hélouvry et al. 1994, Olsson et al. 1998]. Static friction introduces a dead zone in which small inputs produce no motion, and classical approaches often use explicit friction models, compensation terms, or feedback design, including models such as LuGre [Canudas de Wit et al. 1995]. The present work does not replace these methods; it uses dry friction with stiction as an interpretable benchmark for studying whether a conditional generative prior can provide useful open-loop candidate sequences.

3. Method

3.1. Action Diffusion Model for Control Sequences

The proposed Action Diffusion model learns a conditional distribution over control sequences of horizon H,

$$p_\theta(u_{0:H-1} \mid x_0, x^\star)\,, \tag{1}$$

where x_0 is the initial state, $x^\star$ is the desired target state, and $u_{0:H-1}$ is a candidate open-loop control sequence. Training data are generated synthetically. Initial states are sampled from a bounded range, candidate control sequences are sampled from the training control prior described in Sect. 4.2, and the known dynamics are rolled forward to obtain the terminal state x_H. The conditioning vector is

$$c = [x_0^\top, x_H^\top]^\top \in \mathbb{R}^4. \tag{2}$$

The denoising network is then trained to model control sequences consistent with the transition from x_0 to x_H.

3.2. Training Objective

The model is trained with the standard noise-prediction objective used in diffusion models [Ho et al. 2020]. Given a clean control sequence u_0, a diffusion time index t, and Gaussian noise ϵ, a noisy sequence is formed as

$$u_t = \sqrt{\bar{\alpha}_t}\, u_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \tag{3}$$

where β_t is the variance increment of the forward diffusion schedule, $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. In the experiments, the sequence of β_t values is induced by a cosine noise

Table 1. Conditional 1D U-Net used to denoise control sequences of length $H = 64$. All residual blocks use GroupNorm, SiLU activations, and additive conditioning from time and state embeddings.

Stage	Layer(s)	Output shape
Input	Control sequence reshaped to 1 channel	1×64
Embeddings	Sinusoidal time embedding $\rightarrow$ MLP; condition MLP	$64, 64$
Encoder	Conv1D $(1 \rightarrow 64)$, kernel 3	64×64
	ResBlock1D	64×64
	ResBlock1D	64×64
Downsampling	Conv1D, kernel 4, stride 2	64×32
Bottleneck	ResBlock1D	64×32
	ResBlock1D	64×32
Upsampling	Transposed Conv1D, kernel 4, stride 2	64×64
Decoder	Skip addition from encoder, ResBlock1D	64×64
	ResBlock1D	64×64
Output	Conv1D $(64 \rightarrow 1)$, kernel 3	1×64

schedule. The denoiser predicts the injected noise and is trained with mean-squared error:

$$\mathcal{L}(\theta) = \mathbb{E}_{u_0,c,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(u_t, t, c)\|_2^2\right]. \tag{4}$$

The implementation adopts classifier-free guidance [Ho and Salimans 2022] by randomly dropping the conditioning vector c during training with probability $p_{\text{drop}} = 0.2$. During sampling, the conditional prediction uses the specified initial and target states, whereas the unconditional prediction is obtained by replacing the condition with the null condition learned during these dropped-condition training examples. The two predictions are then combined as described below.

3.3. Model Architecture

Each scalar control sequence $u_{0:H-1} \in \mathbb{R}^H$ is represented as a tensor with one channel and H temporal positions, i.e., shape $1 \times H$ with $H = 64$ in the experiments. The denoiser is a compact 1D conditional U-Net that maps this noisy one-channel temporal signal to a noise prediction with the same shape. A sinusoidal embedding encodes the diffusion time step, while the conditioning vector $[x_0, x_H]$ is mapped through a small MLP and injected additively into each residual block. Table 1 summarizes the implemented architecture used in the experiments.

3.4. Sampling with DDIM and Classifier-Free Guidance

At test time, control sequences are sampled with a deterministic DDIM-style reverse process [Song et al. 2021]. Starting from Gaussian noise, the sampler predicts the denoised control sequence over a reduced subset of diffusion steps. For each reverse step, the implementation computes unconditional and conditional noise predictions and combines them as

$$\hat{\epsilon}_{\text{cfg}} = \hat{\epsilon}_{\text{uncond}} + s\left(\hat{\epsilon}_{\text{cond}} - \hat{\epsilon}_{\text{uncond}}\right), \tag{5}$$

where s is the guidance scale. Here, $\hat{\epsilon}_{\text{cond}}$ denotes the prediction with the actual condition $c = [x_0^\top, x^{\star\top}]^\top$, while $\hat{\epsilon}_{\text{uncond}}$ denotes the prediction under the null condition. The predicted clean sequence is clamped during sampling for numerical stability.

3.5. Open-Loop Control

For a target state $x^\star$, each method generates K candidate open-loop sequences. The candidates are rolled out through the known dynamics, and the best sequence is selected by terminal state error,

$$e_{\text{final}} = \|x_H - x^\star\|_2. \tag{6}$$

The same selection rule is used for Action Diffusion, dataset-prior random shooting, uniform random shooting, and for ranking samples within each CEM iteration.

For a multi-step reference, we use segmented open-loop planning. Given a sequence of target states $(x_1^\star, \ldots, x_M^\star)$, the method generates a sequence from the current state to $x_1^\star$, executes the selected segment, then repeats the procedure from the reached state to $x_2^\star$, and so on. This is not an online feedback controller: it is a piecewise composition of state-to-state open-loop plans.

4. Experimental Setup

4.1. Benchmark: Dry Friction with Stiction

We evaluate the approach on a one-dimensional point-mass system with state $x = [p, v]^\top$ and scalar control u. The dynamics combine Coulomb friction, viscous damping, and a static-friction dead zone:

$$\dot{p} = v, \tag{7}$$

$$\dot{v} = \begin{cases} 0, & |v| < \epsilon_v \text{ and } |u| \le f_s, \\ u - f_c \operatorname{sgn}(u), & |v| < \epsilon_v \text{ and } |u| > f_s, \\ u - f_c \operatorname{sgn}(v) - bv, & |v| \ge \epsilon_v. \end{cases} \tag{8}$$

The first case models stiction: when the velocity is close to zero and the input does not exceed the static-friction threshold, the mass remains at rest. The system is simulated with a semi-implicit Euler update, using the post-friction velocity to update the position.

4.2. Dataset Collection

The dataset consists of synthetic pairs $(u_{0:H-1}, c)$, where $c = [x_0^\top, x_H^\top]^\top$. Initial states are sampled uniformly from the configured ranges. No measurement noise or process noise is injected in the current benchmark; every terminal state is obtained by rolling out the known dry-friction dynamics.

For the dry-friction benchmark, the control sequence distribution is intentionally not uniform. Instead, we use a structured prior designed to contain threshold-crossing segments that can initiate motion under stiction. Each sequence is composed of eight constant segments. For each segment, one of three modes is sampled: a high-amplitude "kick" mode with magnitude uniformly sampled in $[0.75u_{\max}, u_{\max}]$ and random sign; a medium-amplitude mode sampled from $[-0.6u_{\max}, 0.6u_{\max}]$; and a near-zero mode sampled from $[-0.15u_{\max}, 0.15u_{\max}]$. The segment values are expanded to length H and smoothed by a first-order filter with coefficient $\alpha = 0.3$. The final sequence is clipped to $[-u_{\max}, u_{\max}]$.

The role of the learned model is therefore to condition this structured prior on the transition from x_0 to x_H, assigning probability to sequences with suitable sign, amplitude, and timing. The dataset-prior random shooting baseline uses the same control prior but does not condition samples on the target state.

Table 2. Main parameter configuration used for training and evaluation.

Component	Value
Horizon H	64
Sampling time Δt	0.05
Control bound $u_{\max}$	0.9
Static friction f_s	0.7
Coulomb friction f_c	0.6
Viscous damping b	0.05
Velocity threshold ϵ_v	10^{-3}
Initial-state range	$p_0 \in [-1, 1]$, $v_0 \in [-0.5, 0.5]$
Training / validation samples	50,000 / 10,000
Diffusion steps T	200
DDIM steps	50
Noise schedule	Cosine
Batch size	256
Training steps	15,000
Learning rate	2×10^{-3}
Condition drop probability p_{drop}	0.2
Base channels / time dim / condition dim	64 / 64 / 64
EMA decay	0.995
Guidance scale	2.0
Multi-step reference sequence	$(0.4, 0)$, $(-0.4, 0)$, $(0.2, 0)$

4.3. Parameter Configuration

Table 2 summarizes the main configuration used in the experiments. The Action Diffusion model was trained on $50{,}000$ training transitions and $10{,}000$ validation transitions. Training was run for $15{,}000$ gradient steps using Adam and an exponential moving average of the model weights.

4.4. CEM Baseline

In addition to random shooting baselines, we evaluate the Cross-Entropy Method (CEM) as a stronger sampling-based optimizer. For a given target state, CEM maintains a Gaussian distribution over control sequences and iteratively refines its mean and variance using the elite candidates with lowest terminal error. At each iteration, candidate sequences are sampled, clipped to the admissible control range $[-u_{\max}, u_{\max}]$, rolled out through the dry-friction dynamics, and ranked according to the terminal error in Eq. (6). The updated sampling distribution is then fitted to the elite set.

To assess the trade-off between optimization quality and computational cost, we vary the number of CEM iterations while keeping the candidate population per iteration fixed within each operating point. We then compare the resulting terminal error against the wall-clock compute time required by CEM and by Action Diffusion sampling. The comparison is performed for two candidate budgets, $K = 32$ and $K = 512$, under the same reference and initial condition.

4.5. Evaluation Metrics

For a generated sequence, we report terminal error e_{final} as defined in Eq. (6). We also report control energy,

$$E_u = \sum_{k=0}^{N-1} u_k^2, \tag{9}$$

and control smoothness,

$$S_u = \sum_{k=0}^{N-2} (u_{k+1} - u_k)^2. \tag{10}$$

For the dry-friction system, we additionally report the number of stuck steps,

$$N_{\text{stuck}} = \sum_k \mathbf{1}\{|v_k| < \epsilon_v,\ |u_k| < f_s,\ |p_k - p_k^\star| > \delta\}, \tag{11}$$

with $\delta = 0.01$. This metric counts steps in which the system is effectively stopped, the applied input is below the static-friction threshold, and the position is still away from the desired reference. For multi-step experiments, stuck steps and computation time are computed over the composed open-loop trajectory. We report both the final error, measured only with respect to the last target after all segments have been executed, and the mean end error, defined as the average terminal error over all segment endpoints.

5. Results

5.1. Single-Target Sample Budget

We begin with a small target displacement, $x^\star = (0.2, 0)$, which is particularly challenging due to the presence of static friction. In this regime, the controller must generate sufficient force to overcome stiction while avoiding excessive overshoot. As a result, successful behavior depends on producing temporally coordinated control sequences, rather than simply low-energy inputs.

Figure 1 provides a qualitative illustration of Action Diffusion open-loop control with $K = 512$ sampled trajectories. Each faint trajectory corresponds to a candidate control sequence, while the highlighted trajectory represents the selected best sample. The figure shows that the Action Diffusion proposal concentrates many candidates near trajectories that approach the target. In particular, the selected trajectory exhibits a clear threshold-crossing and settling pattern: the control signal initially exceeds the static-friction threshold to initiate motion and is subsequently modulated to regulate the system near the target. Since the training prior already contains high-amplitude and near-zero segments, this behavior should be interpreted as conditional selection and timing of available primitives, rather than as discovery of the primitive structure from scratch.

To quantify this behavior, for each method and sample budget $K \in \{16, 32, 128, 512\}$, we perform 10 independent runs. In each run, K candidate control sequences are sampled, and the best sequence is selected according to the terminal error.

Figure 2 reports the resulting terminal error, control energy, and number of stuck steps. Action Diffusion achieves low terminal error even for small K, indicating that the learned conditional distribution concentrates probability mass on controls that are appropriate for the requested transition. Dataset-prior random shooting samples from the same

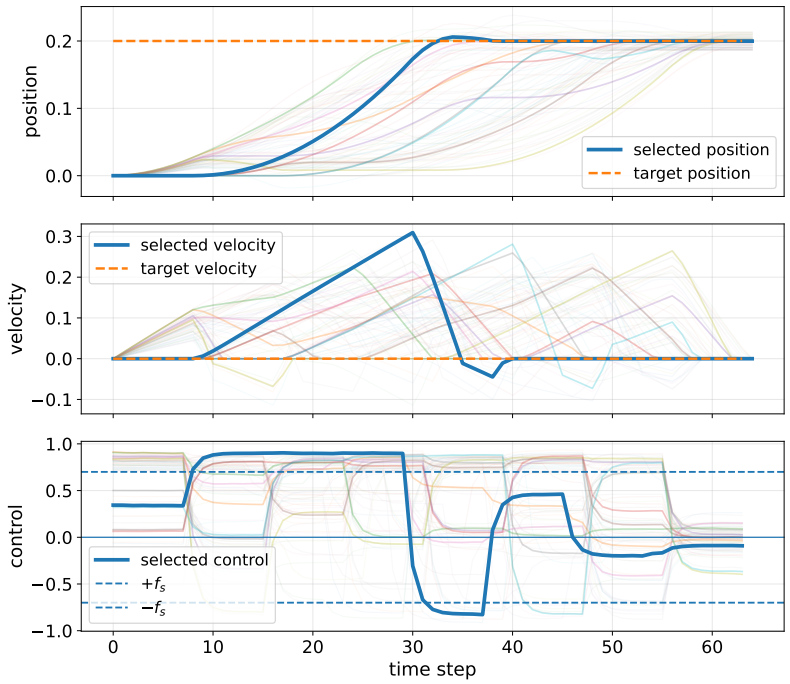


Figure 1. Action Diffusion open-loop control for the small target $x^\star = (0.2, 0)$ using $K = 512$ sampled trajectories. Faint lines represent candidate trajectories, and the highlighted line corresponds to the selected best sequence.

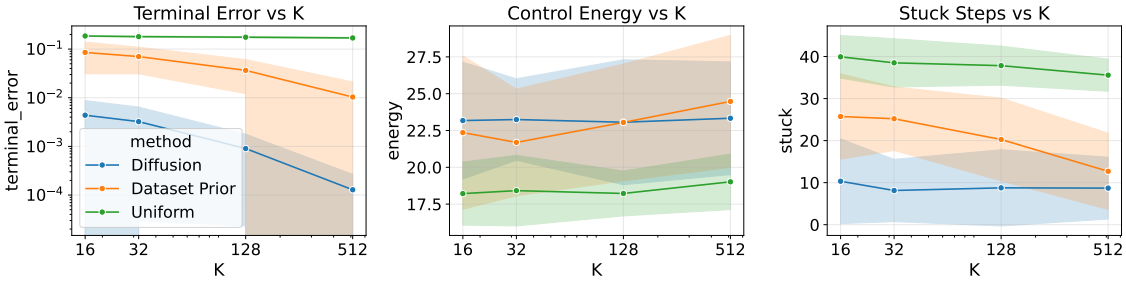


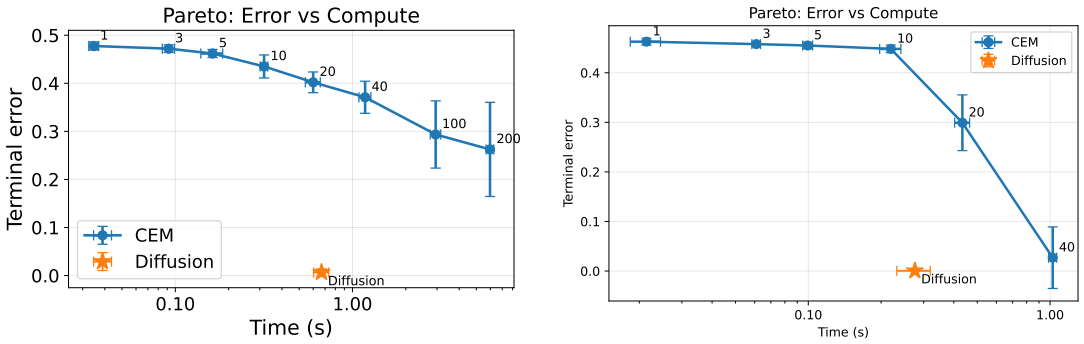
Figure 2. Performance as a function of the sample budget $K \in \{16, 32, 128, 512\}$. Results show mean and standard deviation over 30 independent runs.

structured prior, but because it is not conditioned on the target, many samples contain kicks with unsuitable sign, duration, or timing. Uniform random shooting remains largely ineffective across all values of K because most randomly sampled sequences either fail to cross the static-friction threshold at the right time or produce poorly coordinated motion.

The energy results further highlight that low energy alone is not sufficient for successful control. In particular, a method can have low control energy simply because it does not move the system effectively. For this reason, terminal error and stuck steps are more informative than energy alone in this benchmark.

5.2. Error–Compute Trade-off Against CEM

We also compare Action Diffusion with CEM under a fixed target-setting of $(0.5, 0)$ and initial condition $(0, 0)$, focusing on the trade-off between terminal accuracy and computation time. Figure 3 reports representative low- and high-budget cases, $K = 32$ and



(a) K=32 (b) K=512

Figure 3. Error–compute comparison between Action Diffusion and CEM for representative low- and high-budget regimes. CEM points correspond to different numbers of optimization iterations. Action Diffusion is shown as a single operating point. Lower and leftward points are preferable.

$K = 512$, where thirty runs are computed per configuration, with the resulting mean and standard deviation shown in the plot. For CEM, each point corresponds to a different number of optimization iterations, while Action Diffusion is shown as a single operating point. The comparison shows that the advantage of Action Diffusion depends on the operating regime. For $K = 32$, Action Diffusion achieves lower terminal error at a smaller compute budget than CEM. For $K = 512$, the gap becomes narrower: CEM improves with additional refinement, and the advantage of Action Diffusion is less pronounced.

5.3. Segmented Multi-Step Open-Loop Planning

We next evaluate the segmented open-loop task with the step-reference sequence $(0.4, 0)$, $(-0.4, 0)$, and $(0.2, 0)$. In each segment, Action Diffusion samples candidate control sequences conditioned on the current state and the next target, rolls them out through the known dynamics, and selects the best one by terminal error.

Figure 4 compares Action Diffusion, dataset-prior random shooting, and CEM for $K = 16$ and $K = 512$. For each method and value of K, five independent selected trajectories are plotted, with the thick line showing their mean. With $K = 16$, Action Diffusion follows the intended sequence more consistently, whereas CEM remains almost flat and fails to initiate the required motion. This illustrates the difficulty of iterative refinement when the initial sampled population contains few effective threshold-crossing controls. With $K = 512$, CEM improves on average, but several individual runs still deviate substantially from the reference, indicating high variability in the non-smooth stiction landscape. Action Diffusion produces more consistent threshold-crossing behavior across runs because its proposal distribution is conditioned on the desired state transition.

Quantitative results. Table 3 summarizes the randomized multi-step evaluation over 30 reference sequences and 5 independent sampling runs per sequence. Action Diffusion obtains the lowest final error, mean end error, and number of stuck steps for all values of K. Dataset-prior sampling is faster because it only samples and rolls out candidate controls, but its larger errors show that the unconditioned structured prior is not sufficient. CEM improves as the population size increases, but it requires substantially more computation

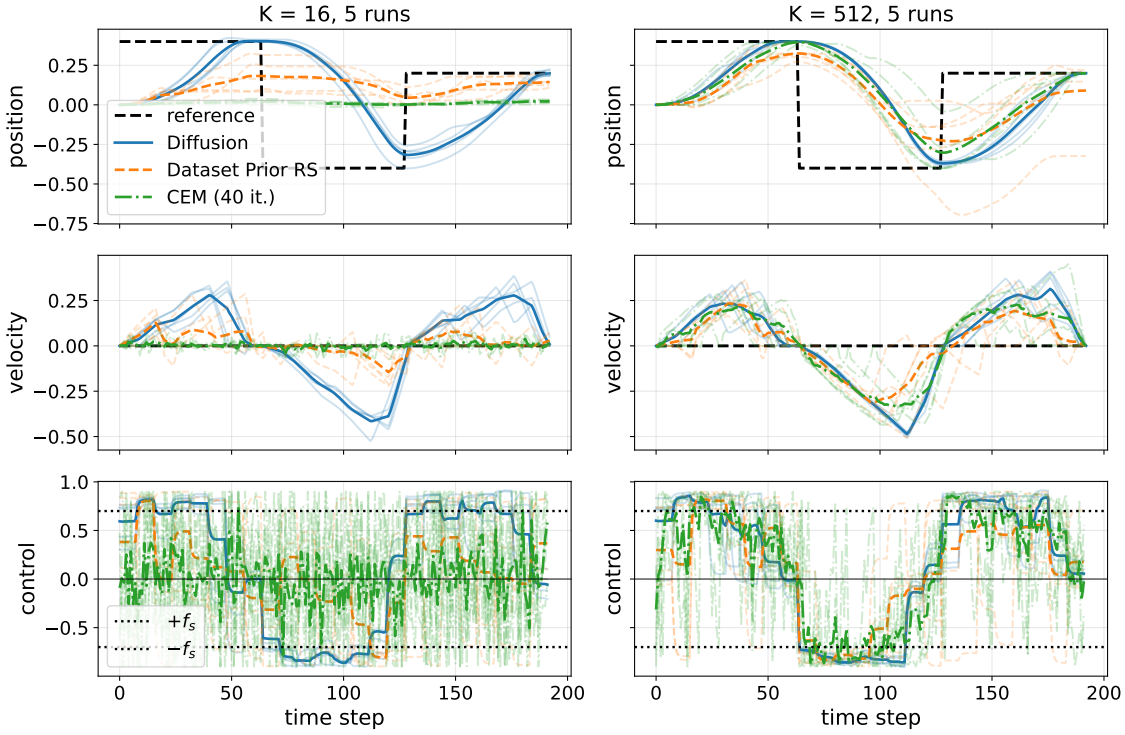


Figure 4. Segmented open-loop trajectories for the multi-step control task under dry friction. Results are shown for $K = 16$ and $K = 512$. For each method and value of K, 5 independent runs are plotted with thin transparent lines, while the mean of them is given by a solid thick line.

under the 40-iteration setting. To test whether the conclusions are specific to the nominal

Table 3. Comparison of methods on multi-step open-loop control. Results are reported as mean ± standard deviation over 30 randomly sampled multi-step reference sequences and 5 independent sampling runs per sequence, totaling 150 evaluations per method and sample budget K.

Method	K	Final Error ↓	Mean End Error ↓	Stuck Steps ↓	Time (s) ↓
Action Diffusion	16	**0.20 ± 0.3**	**0.17 ± 0.1**	**23.2 ± 18.7**	0.68 ± 0.1
Dataset Prior	16	0.38 ± 0.4	0.40 ± 0.2	69.9 ± 19.3	**0.07 ± 0.0**
CEM (40 it.)	16	0.43 ± 0.3	0.49 ± 0.2	105.6 ± 11.3	2.54 ± 0.2
Action Diffusion	32	**0.19 ± 0.3**	**0.16 ± 0.1**	**18.8 ± 18.1**	0.75 ± 0.1
Dataset Prior	32	0.34 ± 0.4	0.37 ± 0.2	60.7 ± 17.6	**0.08 ± 0.0**
CEM (40 it.)	32	0.36 ± 0.3	0.40 ± 0.2	63.0 ± 9.3	2.57 ± 0.3
Action Diffusion	128	**0.18 ± 0.3**	**0.15 ± 0.1**	**17.1 ± 14.9**	0.76 ± 0.1
Dataset Prior	128	0.32 ± 0.4	0.31 ± 0.2	46.5 ± 16.6	**0.11 ± 0.0**
CEM (40 it.)	128	0.31 ± 0.4	0.25 ± 0.2	27.9 ± 16.9	2.56 ± 0.1
Action Diffusion	512	**0.17 ± 0.3**	**0.13 ± 0.1**	**15.9 ± 17.0**	1.56 ± 0.6
Dataset Prior	512	0.29 ± 0.4	0.25 ± 0.2	33.6 ± 15.7	**0.33 ± 0.2**
CEM (40 it.)	512	0.19 ± 0.3	0.14 ± 0.1	22.2 ± 16.7	3.47 ± 1.2

dry-friction configuration, we also repeated the randomized multi-step evaluation for different values of the static-friction threshold (f_s). Table 4 reports the mean end error for

Table 4. Sensitivity to the static-friction threshold f_s. Results report the Mean End Error (as in Table 3), for sample budget $K = 128$.

Method	$f_s = 0.6$	$f_s = 0.7$	$f_s = 0.8$
Action Diffusion	$\mathbf{0.14 \pm 0.1}$	$\mathbf{0.15 \pm 0.1}$	$\mathbf{0.10 \pm 0.1}$
Dataset Prior	0.30 ± 0.2	0.31 ± 0.2	0.31 ± 0.2
CEM (40 it.)	0.22 ± 0.2	0.25 ± 0.2	0.22 ± 0.2

($f_s \in 0.6, 0.7, 0.8$) using ($K = 128$). Action Diffusion is trained from scratch for each value of f_s. It achieves the lowest mean end error in all three settings, suggesting that the benefit of conditioning the action-sequence prior is not restricted to a single stiction threshold. The experiment is not intended as a complete robustness analysis, since the remaining system parameters are kept fixed, but it provides an initial sensitivity check over the main parameter governing the stiction dead zone.

6. Discussion

The results suggest that Action Diffusion is useful as a structured open-loop proposal distribution for non-smooth control. The comparison with dataset-prior random shooting is central: both methods use the same underlying control-sequence prior, but only Action Diffusion conditions it on the desired transition. Thus, the gains should be attributed not to the mere presence of high-amplitude "kick" segments in the data, but to the model's ability to assign higher probability to sequences with suitable direction, magnitude, and timing. The CEM comparison further indicates that this advantage is strongest in low-budget regimes, where iterative refinement from a generic sampling distribution is inefficient due to flat or ineffective regions induced by stiction. At larger candidate budgets, CEM improves and the advantage of Action Diffusion becomes less pronounced.

7. Conclusion

This paper studied Action Diffusion as an open-loop control-sequence proposal distribution for a point-mass system with dry friction and stiction. The benchmark requires inputs that cross a static-friction threshold to initiate motion, making naive sampling inefficient. The results show that Action Diffusion can achieve lower terminal error and fewer stuck steps than uniform and dataset-prior random shooting, especially with small sample budgets. The CEM comparison indicates that this advantage is strongest in low-budget regimes, while iterative optimization can reduce the gap when more computation is available. Qualitative results show interpretable threshold-crossing, motion, and settling phases, but these should be understood as conditional recombinations of structured segments already present in the training prior. Future work will consider richer mechanical systems, stochastic disturbances, additional learning baselines, and receding-horizon feedback execution.

Acknowledgments. This work was partially funded by the National Council for Scientific and Technological Development – CNPq, Brazil (Grant No. 420148/2025-6), and FAPESC – Foundation for Research and Innovation Support of the State of Santa Catarina (Grant No. 2024TR000090).

Disclosure of Interests. The author has no competing interests to declare that are relevant to the content of this article.

References

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. (2023). Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations*.

Armstrong-Hélouvry, B., Dupont, P., and Canudas de Wit, C. (1994). A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30(7):1083–1138.

Canudas de Wit, C., Olsson, H., Åström, K. J., and Lischinsky, P. (1995). A new model for control of systems with friction. *IEEE Transactions on Automatic Control*, 40(3):419–425.

Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. (2023). Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851.

Ho, J. and Salimans, T. (2022). Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. (2022). Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9902–9915.

Kemmer, B., Simões, R., Ivamoto, V., and Lima, C. (2023). Performance analysis of generative adversarial networks and diffusion models. In *Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS)*. Springer.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives. *arXiv preprint arXiv:2005.01643*.

Machado, B. M. and Antonelo, E. A. (2025). Investigating diffusion models for offline behavior cloning in autonomous driving scenarios. In *2025 Brazilian Conference on Robotics (CROS)*, volume 1, pages 1–6. IEEE.

Olsson, H., Åström, K. J., Canudas de Wit, C., Gäfvert, M., and Lischinsky, P. (1998). Friction models and friction compensation. *European Journal of Control*, 4(3):176–195.

Rubinstein, R. Y. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190.

Song, J., Meng, C., and Ermon, S. (2021). Denoising diffusion implicit models. In *International Conference on Learning Representations*.

Williams, G., Drews, P., Goldfain, B., Rehg, J., and Theodorou, E. (2017). Information-theoretic model predictive control: Theory and applications to autonomous driving. In *ICRA*.