ASAP: ARBITRARY-STEP SAFE PLANNING WITH MEANFLOW AND CONTROL BARRIER FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative models have emerged as a powerful paradigm for data-driven robotic planning. Existing flow-matching planners typically rely on multi-step sampling and lack principled mechanisms for enforcing safety constraints at test time. Bridging these limitations is essential for real-world deployment, where planners must support both fast generation and formal safety guarantees. We propose **A**rbitrary-Step **SA**fe **P**lanning (ASAP), a control-augmented MeanFlow framework for safe generation with any number of sampling steps, including one step. ASAP introduces a virtual control input governed by an affine barrier constraint that accounts for the sampling step size, yielding formal safety guarantees of generation for any number of steps. Without retraining, ASAP enforces unseen test-time constraints across navigation, locomotion, manipulation, and dexterous manipulation benchmarks, while planning at least $20\times$ faster than safety-guaranteed baselines on navigation, locomotion, and manipulation, and $5\times$ faster on dexterous manipulation. We further validate ASAP on a real-world robot handover task, where it maintains safety and high task performance under an obstacle, requiring only $0.04\,\mathrm{s}$ per plan and achieving over a $200\times$ speedup over the safe baseline.

1 INTRODUCTION

Generative models, such as diffusion models (Ho et al., 2020) and flow matching (Lipman et al., 2022), have recently emerged as a powerful paradigm for robotic trajectory planning. By learning trajectory distributions directly from data, they can naturally capture complex and multimodal behavior (Chi et al., 2025) in long-horizon tasks (Janner et al., 2022). Unlike conventional trajectory-optimization planners (Kalakrishnan et al., 2011; Posa et al., 2014), which rely on iterative online optimization and accurately learned system models, generative approaches (Janner et al., 2022; Zheng et al., 2023) implicitly encode system dynamics and directly generate full trajectories at inference time. This enables scalable modeling of high-dimensional behaviors, supports task compositionality, and mitigates error accumulation from recursive rollout.

Despite their promise, existing generative planning approaches typically suffer from one or both of two key limitations: slow multi-step inference and a lack of formal safety guarantees, as summarized in Table 1. Rejection sampling discards unsafe trajectories but becomes ineffective when safe samples are rare under unseen constraints. Post-processing methods (Mazé & Ahmed, 2023) correct trajectories after generation but can substantially deviate from the learned distribution (Römer et al., 2024). Guidance-based approaches (Kondo et al., 2024; Dhariwal & Nichol, 2021; Mizuta & Leung, 2024) steer samples toward safe regions, yet soft guidance may leave residual violations without formal guarantees. To obtain certified safety, recent methods regulate the generative dynamics of diffusion models (Xiao et al., 2025) or flow models (Huang et al., 2026; Yang et al., 2026) using control-theoretic mechanisms. However, these guarantees require repeated safety enforcement throughout multi-step generation, introducing substantial sampling overhead and limiting real-time planning.

To accelerate planning, recent works exploit the self-consistency property of generative models (Frans et al., 2025; Song et al., 2023) to enable arbitrary-step sampling, including one-step generation. MeanFlow (Geng et al., 2025) further learns the average velocity over arbitrary time intervals, enabling efficient trajectory generation in robot manipulation (Sheng et al., 2026) and autonomous driving (Wang et al., 2026). However, these methods lack formal safety guarantees. The safety mechanisms above are derived in continuous time (Xiao et al., 2025; Huang et al., 2026), so their

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.

guarantees do not directly extend to finite-step generation, particularly under few-step sampling with larger discretization errors. Consequently, guaranteeing safety under arbitrary-step generation remains an open problem, limiting its potential for real-time safe planning.

To address these limitations, we propose **A**rbitrary-Step **SA**fe **P**lanning (ASAP), a control-augmented MeanFlow framework for safe trajectory generation with any number of sampling steps. Guaranteeing safety for arbitrary-step generation is non-trivial because each MeanFlow update may span a large interval, so safety at the current sample does not generally imply safety at the next. ASAP directly constrains the next generated sample using an arbitrary-step affine barrier that approximates this discrete safety condition. Exploiting signed-distance geometry with a minimum-norm correction makes this approximation accurate, yielding zero conservatism. The resulting control is computed through a lightweight quadratic program, enabling safe planning with only a few, or even one, model inference and control solve, without retraining. As summarized in Table 1, to our knowledge, ASAP is the first generative planner with formal safety guarantees for arbitrary-step generation, including the one-step case. We evaluate ASAP across navigation, locomotion, and manipulation benchmarks, where it satisfies all safety constraints while planning at least $20\times$ faster than safety-guaranteed baselines and over $200\times$ faster on a real-world robot-handing task, requiring only $0.04\,\mathrm{s}$ per plan. To evaluate scalability, we further test on the high-dimensional Adroit task, ASAP remains safe while planning over $5\times$ faster than safety-guaranteed baselines.

Table 1. Methods comparison. Only ASAP ensures safety, arbitrary-step generation, and a formal guarantee.

Generative Planners	Safety constraint	Arbitrary step	Formal guarantee
SafeFlowMatcher (Yang et al., 2026)	✓	✗	✓
MeanFlow (Geng et al., 2025)	✗	✓	✗
CoBL-diffusion (Mizuta & Leung, 2024)	✓	✗	✗
SAD-Flower (Huang et al., 2026)	✓	✗	✓
ASAP (Ours)	✓	✓	✓

2 Related Work

2.1 Planning with Diffusion-based and flow-based Generative Models

Diffusion models (Ho et al., 2020; Song et al., 2020) and flow-based models (Lipman et al., 2022; Albergo & Vanden-Eijnden, 2022) have been increasingly adopted for data-driven robotic planning to learn complex expert behaviors from offline datasets (Chi et al., 2025; Huang et al., 2024a). Conditioned on the current state, these models can generate state-action trajectories directly (Janner et al., 2022; Zheng et al., 2023) or high-level trajectories tracked by downstream controllers (Chi et al., 2025; Ajay et al., 2022). However, conventional generative planners require iterative multi-step sampling, limiting their computational efficiency for real-time planning. Recent arbitrary-step generative models (Frans et al., 2025; Geng et al., 2025) enable few- or one-step generation and have been applied to robotic manipulation and autonomous driving (Sheng et al., 2026; Wang et al., 2026). Nevertheless, these fast generative planners have no mechanism to guarantee that the generated trajectories satisfy safety constraints, which is the gap addressed by ASAP.

2.2 Constraint-Aware Planning with Generative Models

To incorporate safety constraints into generative planning, existing methods broadly consider training-time and test-time mechanisms. Training-based approaches (Ajay et al., 2022; Hung et al., 2025) incorporate constraint information during learning to increase the likelihood of feasible trajectories. However, they require additional constraint-aware data and often generalize poorly to unseen constraints. Test-time approaches instead enforce constraints during or after generation. Post-processing methods (Li et al., 2024; Mazé & Ahmed, 2023) modify trajectories after sampling, but such corrections are unaware of the learned distribution and may cause substantial distributional deviation. More recent works introduce control-theoretic mechanisms directly into the generation. Constrained optimal-control layers are integrated into diffusion sampling (Römer et al., 2024), while control barrier functions (Ames et al., 2016) have been applied to diffusion-based planners (Xiao et al., 2025) and flow-based planners (Yang et al., 2026; Huang et al., 2026) to provide formal safety guarantees. However, these methods rely on repeated prediction and immediate safety correction throughout multi-step generation, incurring substantial sampling cost and limiting real-time applicability. In contrast, ASAP directly constrains each finite-step transition through a discrete-time barrier condition, providing safety guarantees for arbitrary-step generation, including a single step.

3 PRELIMINARIES AND PROBLEM SETTING

3.1 FLOW MATCHING

Given a trajectory dataset $\mathcal{D}$ collected by an expert planner, continuous-time generative models learn to transport samples from a prior distribution p_0 to the expert trajectory distribution p_1. The transport is defined by a time-dependent vector field $\boldsymbol{v}_t(\boldsymbol{\tau}_t, t)$ through the ordinary differential equation (ODE)

$$\dot{\boldsymbol{\tau}}_t = \boldsymbol{v}_t(\boldsymbol{\tau}_t, t), \qquad t \in [0,1], \qquad \boldsymbol{\tau}_0 \sim p_0, \tag{1}$$

where $\boldsymbol{\tau}_t$ denotes the nominal trajectory generated by the uncontrolled flow in eq. (1). Flow matching (Lipman et al., 2022) can be applied to learn the vector field $\boldsymbol{v}_t(\boldsymbol{\tau}_t, t)$ with conditional flow matching loss $\mathcal{L}_{\text{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\boldsymbol{\tau}_0,\boldsymbol{\tau}_1} ||\boldsymbol{v}_t^{\boldsymbol{\theta}}(\boldsymbol{I}_t, t) - \dot{\boldsymbol{I}}_t||_2^2$, where $\dot{\boldsymbol{I}}_t(\boldsymbol{\tau}_1, \boldsymbol{\tau}_0) = \dot{\alpha}(t)\boldsymbol{\tau}_1 + \dot{\beta}(t)\boldsymbol{\tau}_0$ follows the scheduled interpolation $\boldsymbol{I}_t = \alpha(t)\boldsymbol{\tau}_1 + \beta(t)\boldsymbol{\tau}_0$ with α, β such that $\alpha(0) = 0, \alpha(1) = 1$and $\alpha(t) + \beta(t) = 1$. For trajectory planning, each sample $\boldsymbol{\tau}_1$ represents a complete planned trajectory. Thus, integrating eq. (1) with the learned vector field transports an initial sample $\boldsymbol{\tau}_0 \sim p_0$ to a trajectory $\boldsymbol{\tau}_1 \sim p_1$ that follows the expert distribution.

3.2 PROBLEM STATEMENT

Following the generative process in eq. (1), $\boldsymbol{\tau}_1 = \{\boldsymbol{\tau}_1(0), \ldots, \boldsymbol{\tau}_1(H-1)\}$ denotes the planned trajectory at the end of the process $t = 1$, where $H \in \mathbb{N}$ is the planning horizon and $k = 0, \ldots, H-1$ indexes trajectory states along this horizon, distinct from the generative time $t \in [0, 1]$. For reliable real-world deployment, the planned trajectory along horizon H must remain within a safe set $\mathbb{S}$, e.g., for collision avoidance, which may encode test-time constraints unseen in the training dataset $\mathcal{D}$.

Objective. Given a pretrained generative planner $\boldsymbol{v}_t^{\boldsymbol{\theta}}$, our goal is to generate

$$\boldsymbol{\tau}_0 \sim p_0 \xrightarrow{\boldsymbol{v}_t^{\boldsymbol{\theta}}} \boldsymbol{\tau}_1 \sim p_1, \qquad \boldsymbol{\tau}_1(k) \in \mathbb{S}, \quad \forall k = 0, \ldots, H-1. \tag{2}$$

In particular, we require the planner to (i) generate trajectories efficiently with an arbitrary number of sampling steps, (ii) guarantee safety throughout the planned trajectory, and (iii) preserve task-relevant behavior by minimally deviates from the expert trajectory distribution p_1.

4 METHOD

Existing safe flow planners establish safety by continuously enforcing control barrier function (CBF) conditions along the generative ODE. Such continuous-time guarantees do not directly extend to arbitrary-step sampling: each update spans a finite interval, so an instantaneous CBF condition may not ensure next-step barrier satisfaction. To address this gap, we propose ASAP, which starts from a discrete-time barrier formulation and derives an affine barrier condition guaranteeing safety for arbitrary sampling steps N, including $N = 1$. In Section 4.1, we establish the control-augmented flow and its continuous-time safety guarantee, providing the foundation for arbitrary-step safety. In Section 4.2, we derive an arbitrary-step affine barrier constraint from the discrete-time safety condition and establish formal safety guarantees, while Section 4.3 analyzes the approximation induced by the affine barrier and establishes zero conservatism for the SDF formulation.

4.1 CONTROL-AUGMENTED GENERATIVE MODEL PLANNING

To ensure satisfaction of safety for the generation process, we augment the nominal flow in eq. (1) at test time with a control input $\boldsymbol{u}_t$, yielding a controlled flow process

$$\dot{\boldsymbol{\tau}}_t = \frac{d}{dt}\boldsymbol{\tau}_t = \boldsymbol{v}_t(\boldsymbol{\tau}_t, t) + \boldsymbol{u}_t, \quad t \in [0,1], \quad \boldsymbol{\tau}_0 \sim p_0, \tag{3}$$

where $\boldsymbol{u}_t = 0$ recovers the nominal flow generation process as in eq. (1). The formulation in eq. (3) allows us to reinterpret the requirement in eq. (2) as a control design problem: design $\boldsymbol{u}_t$ for test-time generation so that the planned trajectories satisfy the safety constraints.

Control Barrier Constraints for Safety. We formalize this control objective using control barrier functions (CBFs) (Ames et al., 2016), applied to each trajectory state $\boldsymbol{\tau}_t(k)$, $k = 0, \ldots, H-1$ to

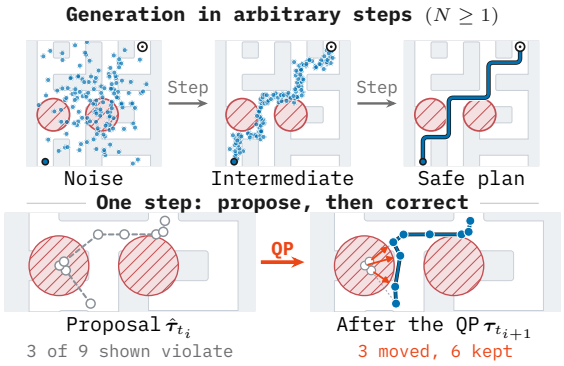

Figure 1. Schematic overview of ASAP. *Top:* $\tau_{t_0} \sim p_0$ becomes a safe plan in any $N \geq 1$ steps. *Bottom:* the QP minimally corrects MeanFlow's H-state proposal, restoring the barrier for constraints unseen in training (red).

Algorithm 1. Planning by ASAP.

Input: $\mu^{\theta}_{t_i}$, steps N, barrier h, ρ_{t_i} by eq. (10)

$\tau_{t_0=0} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, $\Delta t = 1/N$
for $i = 0, \cdots, N-1$ **do**
  $\hat{\tau}_{t_i} = \tau_{t_i} + \mu^{\theta}_{t_i} \Delta t$ eq. (8)
  $u_{t_i} \leftarrow \mathrm{QP}(\hat{\tau}_{t_i}, \rho_{t_{i+1}})$ eq. (13)
  $\tau_{t_{i+1}} = \hat{\tau}_{t_i} + u_{t_i} \Delta t$ eq. (9)
end for

Output: $\tau_{t_{N=1}}$, with $\tau_1(k) \in \mathbb{S}$ for all k

Cost per step: one forward pass and one QP.
Guarantee: Theorem 4.1 — the returned τ_1 satisfies every constraint, for every N.

enforce trajectory-level safety. Under the controlled dynamics in eq. (3), the CBF condition prevents the barrier value from decreasing too rapidly near the safety boundary:

$$h(\tau_t(k)) + \dot{h}(\tau_t(k)) \geq 0, \quad \dot{h}(\tau_t(k)) = \nabla h(\tau_t(k))^{\top}(v_t(\tau_t(k), t) + u_t), \tag{4}$$

for all $k = 0, \ldots, H-1$. Since the barrier constraint is enforced independently at each trajectory index k, we use $h(\tau_t) := h(\tau_t(k))$ in the rest of the paper for simplicity.

Time-Varying Relaxation of CBFs. Although CBFs have been widely applied in dynamic systems, their safety condition can be relaxed for flow dynamics eq. (3) since safety is only required at the terminal time $t=1$. To avoid unnecessarily constraining intermediate generative states, we introduce a time-varying relaxed schedule $\rho_t = -\bar{h}(1-\alpha_t)$, where $\alpha_t : [0,1] \to [0,1]$ is nondecreasing with $\alpha_0 = 0, \alpha_1 = 1$, and $\bar{h} \in \mathbb{R}_+$ is chosen such that $\bar{h} \geq \max\{-h(\tau_0), 0\}$. We formulate the minimum-norm optimal control problem with the time-varying CBF condition to solve for u^*_t

$$u^*_t := \arg\min_{u_t \in \mathcal{F}_t} \|u_t\|^2, \qquad \mathcal{F}_t := \left\{ u_t \,\middle|\, h(\tau_t) + \dot{h}(\tau_t) \geq \rho_t + \dot{\rho}_t \right\} \text{ for all } t \in [0,1], \tag{5}$$

while minimally perturbing the original vector field $v_t(\cdot, \cdot)$. The condition $\mathcal{F}_t$ in eq. (5) is most relaxed at $t=0$ and tightens to the safety requirement at $t=1$. This gives safety guarantees. [1]

Lemma 4.1 (Continuous-time Safe Flow). *If the CBF h is differentiable at τ_t, applying the control u_t in eq.* (5) *to eq.* (3) *guarantees safety at $t = 1$ for every initial condition τ_0.*

Lemma 4.1 establishes safe generation for our control-augmented flow with the time-varying relaxation ρ_t. Extending this guarantee to arbitrary-step generation is non-trivial: safety relies on immediate correction, whereas large sampling intervals delay the next intervention. Thus, a continuous-time CBF condition may not ensure next-step safety if the step size is large. This motivates the discrete-time reformulation in Section 4.2, which directly constrains the next generated sample.

4.2 Arbitrary-Step Flow with Safety

To enable generation with arbitrary steps, we first consider the uncontrolled flow in eq. (1) and adopt MeanFlow (Geng et al., 2025) to learn the nominal average velocity μ over a finite interval $[r,t]$ by $\mu(\tau_t, r, t) \triangleq \frac{1}{t-r} \int_r^t v(\tau_z, z) dz$. Differentiating it yields the MeanFlow identity $\mu(\tau_t, r, t) = v(\tau_t, t) - (t-r)(v(\tau_t, t)\partial_{\tau}\mu + \partial_t \mu)$, which allows us to learn the interval-averaged velocity $\mu(\tau_t, r, t)$ that summarizes the flow over $[r,t]$, replacing the repeated instantaneous-velocity $v(\tau_t, t)$ predictions of standard flow matching in eq. (1):

$$\mathcal{L}(\theta) = \mathbb{E}\|\mu^{\theta}(\tau_t, r, t) - \mathrm{sg}(\mu_{\mathrm{tgt}})\|_2^2 \qquad \text{s.t.} \quad \mu_{\mathrm{tgt}} = v - (t-r)(v\partial_{\tau}\mu^{\theta} + \partial_t \mu^{\theta}), \tag{6}$$

where the sg is the stop-gradient operation. By predicting the nominal average velocity over arbitrary intervals, MeanFlow enables few-step and even one-step trajectory generation via:

$$\tau_t = \tau_r + (t-r)\mu^{\theta}(\tau_r, r, t). \tag{7}$$

[1] For a proof of Lemma 4.1 see Appendix B

Controlled MeanFlow for Arbitrary-Step Generation. To enforce safety during arbitrary-step generation, following eq. (3), we augment MeanFlow update eq. (7) with a control input $\boldsymbol{u}_{t_i}$ that corrects the nominal sample toward constraint satisfaction. An N-step MeanFlow generation partitions $[0,1]$ into N intervals, with $t_i = \frac{i}{N}$, $i = 0, \ldots, N$, and sampling period $\Delta t = \frac{1}{N}$. For the i-th interval $[t_i, t_{i+1}]$, we denote the MeanFlow prediction by $\boldsymbol{\mu}^{\boldsymbol{\theta}}_{t_i}$ and formulate the control-augmented update as

$$\textit{Nominal Update: } \hat{\boldsymbol{\tau}}_{t_i} = \boldsymbol{\tau}_{t_i} + \boldsymbol{\mu}^{\boldsymbol{\theta}}_{t_i} \Delta t \tag{8}$$

$$\textit{Controlled Update: } \boldsymbol{\tau}_{t_{i+1}} = \hat{\boldsymbol{\tau}}_{t_i} + \boldsymbol{u}_{t_i} \Delta t, \quad i = 0, \ldots, N-1, \tag{9}$$

where eq. (8) gives the nominal MeanFlow prediction, while $\boldsymbol{u}_{t_i}$ corrects it toward constraint satisfaction. To progressively tighten the safety requirement across discrete generation steps, the relaxation schedule ρ_t defined in Section 4.1 is evaluated at each sampling time:

$$\rho_{t_i} = -\bar{h}(1 - \alpha_{t_i}), \quad -\bar{h} = \rho_{t_0=0} \leq \rho_{t_1} \leq \cdots \leq \rho_{t_N=1} = 0, \quad \bar{h} \geq \max\{-h(\boldsymbol{\tau}_{t_0}), 0\} \tag{10}$$

Arbitrary-Step Affine Barrier Constraints. For the discrete updates in eq. (8) and eq. (9), safety requires the next generated sample to satisfy $h(\boldsymbol{\tau}_{t_{i+1}}) \geq \rho_{t_{i+1}}$. Since a discrete update may span a large interval Δt, the continuous-time CBF condition enforced at t_i does not generally guarantee barrier satisfaction at t_{i+1}. Therefore, motivated by a discrete-time barrier condition (Agrawal & Sreenath, 2017), we directly constrain the next-step generated sample $\boldsymbol{\tau}_{t_{i+1}}$ from eq. (9) as

$$h(\boldsymbol{\tau}_{t_{i+1}}) = h(\hat{\boldsymbol{\tau}}_{t_i} + \boldsymbol{u}_{t_i} \Delta t) \geq \rho_{t_{i+1}}. \tag{11}$$

Since the discrete barrier condition eq. (11) is generally nonlinear in $\boldsymbol{u}_{t_i}$, we linearize it at the nominal MeanFlow update $\boldsymbol{u}_{t_i} = 0$ and later show that, under our formulation, the linearization introduces no approximation error. This yields our arbitrary-step affine barrier constraint:

$$h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i} \Delta t) \geq \rho_{t_{i+1}}. \tag{12}$$

With the next-step threshold $\rho_{t_{i+1}}$ in eq. (10), we choose the $\boldsymbol{u}_{t_i}$ by the minimum-norm control as:

$$\boldsymbol{u}_{t_i} = \arg\min_{\boldsymbol{u}} ||\boldsymbol{u}||^2 \quad \text{s.t. eq. (12)} \tag{13}$$

Because the barrier constraint is affine in $\boldsymbol{u}_{t_i}$, eq. (13) is a quadratic program (QP) that can be solved efficiently. We construct the CBF as a signed distance function (SDF) (Long et al., 2021; Park et al., 2019), whose value gives the signed distance to the boundary of $\mathbb{S}$. [2]. The key insight of ASAP is to exploit a geometric property of the SDF: at a differentiable point outside $\mathbb{S}$, $||\nabla h|| = 1$, ∇h points toward the nearest safe boundary, and h varies linearly along this direction. When the barrier constraint is active, the minimum-norm solution of eq. (13) is aligned with ∇h, as any orthogonal component increases the control norm without affecting the constraint. Consequently, the barrier value in eq. (12) exactly matches the true next-step value $h(\boldsymbol{\tau}_{t_{i+1}})$. Hence, satisfying the affine constraint in eq. (12) guarantees $h(\boldsymbol{\tau}_{t_{i+1}}) \geq \rho_{t_{i+1}}$, yielding the following results

Theorem 4.1 (**Arbitrary-Step Safe Flow**). *Given an initial condition $\boldsymbol{\tau}_{t_0}$ and if the SDF is differentiable for all $\hat{\boldsymbol{\tau}}_{t_i}$, the control input $\boldsymbol{u}_{t_i}$ in eq.* (13) *ensures that the solution $\boldsymbol{\tau}_{t_N}$ of eq.* (9) *is safe.*[3]

Since Theorem 4.1 holds for arbitrary N, the one-step safety guarantee follows immediately.

Corollary 4.1 (**One-Step Safe Flow**). *Under the conditions of Theorem 4.1, for $N = 1$ and $\Delta t = 1$, the control input $\boldsymbol{u}_{t_0}$ in eq.* (13) *ensures that the one-step solution $\boldsymbol{\tau}_{t_1}$ of eq.* (9) *is safe.*

Our ASAP procedure is summarized in Algorithm 1, and Figure 1 shows one sampling step. With setting $N = 1$, planning requires only a single MeanFlow inference and a single QP solve, enabling real-time safe trajectory generation. Importantly, using an SDF introduces zero conservatism in the affine safety condition for any sampling step size, as analyzed in the next section.

Remark 4.1. *The differentiability assumption is mild since SDFs are differentiable almost everywhere for sets with sufficiently regular boundaries (Gilbarg et al., 1977). At nondifferentiable SDF points, generalized gradients (Long et al., 2026; Glotfelter et al., 2017) can replace ordinary gradients by selecting a valid direction toward a nearest boundary point for the safety correction. Thus, this assumption can be handled with the practical use of SDF-based constraints (Long et al., 2021).*

[2] For a formal definition and more details of the SDF $h(\boldsymbol{\tau}_{t_i})$, see Appendix A

[3] The proof of Theorem 4.1 and Corollary 4.1 are provided in Appendix B

Remark 4.2. *Unlike standard CBF control problems, the control input $\boldsymbol{u}_{t_i}$ in the flow dynamics eq.* (9) *is unconstrained and acts directly on the generated trajectory. Consequently, each affine barrier constraint in eq.* (12) *is always feasible whenever the SDF is differentiable. If $h(\hat{\boldsymbol{\tau}}_{t_i}) \geq \rho_{t_{i+1}}$, $\boldsymbol{u}_{t_i} = \mathbf{0}$ is feasible; otherwise, since a differentiable SDF satisfies $\|\nabla h(\hat{\boldsymbol{\tau}}_{t_i})\| > 0$, a correction along $\nabla h(\hat{\boldsymbol{\tau}}_{t_i})$ can always satisfy the constraint with equality. Thus, feasibility follows directly from the SDF geometry and the integrator-like generation dynamics; see Appendix B.5 for the formal result.*

4.3 Conservatism Analysis

The previous section enforces arbitrary-step safety using the affine barrier condition in eq. (12). Here, conservatism refers to the mismatch between the barrier value used by this condition and the true next-step barrier value, which may induce unnecessary safety correction. For the exact SDF used by ASAP, the minimum-norm correction follows the SDF gradient, along which the SDF varies linearly, eliminating this mismatch for any sampling step size. We further analyze general CBFs in Appendix B.6, where the approximation error is bounded.

Safety Conservatism. With the affine barrier function in eq. (12), the value at next step is approximated by $\hat{h}(\hat{\boldsymbol{\tau}}_{t_i}) = h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i} \Delta t)$, and the resulting $\boldsymbol{u}_{t_i}$ obtained by eq. (13) updates $\hat{\boldsymbol{\tau}}_{t_i} \to \boldsymbol{\tau}_{t_{i+1}}$. We define the total conservatism $\bar{C}$ as the constraint residual summed over the N steps

$$\bar{C} = \sum_{i=0}^{N-1} C_i, \quad C_i = h(\boldsymbol{\tau}_{t_{i+1}}) - \hat{h}(\hat{\boldsymbol{\tau}}_{t_i}), \tag{14}$$

Based on the properties of SDF geometry described in Section 4.2, we can obtain the following result

Theorem 4.2 (**Zero SDF Conservatism**). *If the SDF h is differentiable at $\hat{\boldsymbol{\tau}}_{t_i}$ for all i, then the control input $\boldsymbol{u}_{t_i}$ from eq.* (13) *yields $\bar{C} = 0$ for any number of sampling steps N.* [4]

If the affine barrier constraint is inactive, $\boldsymbol{u}_{t_i} = 0$ and the approximation error is zero. If active, the minimum-norm correction follows the SDF gradient, along which the SDF varies linearly until the boundary. Therefore, the barrier condition remains exact over the finite-step correction, so the predicted and true next-step barrier values coincide, yielding zero conservatism for any N step. This exact approximation avoids unnecessary safety corrections, helping preserve the original plan.

5 Experiments

5.1 Experiment Setup

We evaluate ASAP on a variety of simulation benchmarks with test-time constraints as follows.

- **Maze2d** (Fu et al., 2020) considers point-mass navigation from an initial state to a target location. We introduce two previously unseen obstacles as state constraints while preserving at least one feasible route to the goal. The dataset is generated using a navigation planner in the large maze.
- **Hopper** (Fu et al., 2020) is a locomotion benchmark in which an agent must move forward efficiently. We impose a torso-height constraint, which conflicts with forward locomotion. We use the Med-Expert (ME) dataset, combining expert and partially trained soft actor-critic trajectories.
- **KUKA Block-Stacking** (Janner et al., 2022) is a 7-DoF robotic manipulator tasked with stacking multiple blocks. Joint-space state constraints are introduced to prevent self-collision during manipulation. The dataset is generated with the PDDLStream planner (Garrett et al., 2020).

Evaluation Metrics. We quantify trajectory safety violation (s-vio) by the minimum constraint margin $-\min_k \min\{h_k(\boldsymbol{\tau}_1), 0\}$, where 0 indicates constraint satisfaction and larger values correspond to larger violations. Sampling time is measured as the total planning time (seconds), reflecting the capability for real-time deployment. Task performance is evaluated using normalized total rewards (score) for D4RL tasks, binary success rewards for block stacking following Janner et al. (2022). All results are evaluated on the same hardware, and the settings of baseline sampling steps follow (Xiao et al., 2025) for a fair planning-time comparison. Further details are provided in Appendix D.

Baselines. We compare our proposed ASAP with the following generative planning baselines:

[4] The proof of Theorem 4.2 and the corresponding validation are provided in Appendix B.4 and C.5.

Table 2. Main results. Safety violation (s-vio; 0 means every constraint is satisfied), planning time (s), and task score of every method shown as mean±std on the navigation, locomotion, and manipulation tasks. ASAP satisfies every constraint while planning at least 20× faster than the safety-guaranteed baselines.

Task	Metric	MF	SFM	SAD-F	Diffuser	CG	S-Diff.	CoBL	ASAP
Maze2d (Large)	s-vio (↓)	0.42±0.29	**0.00±0.00**	**0.00±0.00**	0.22±0.28	0.69±0.20	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
	time (↓)	**0.01±0.00**	15.50±0.75	35.84±2.51	1.64±0.03	1.83±0.03	17.08±2.29	2.26±0.03	0.06±0.01
	score (↑)	**1.46±0.47**	1.30±0.67	1.44±0.28	1.38±0.51	0.42±0.29	**1.46±0.23**	0.79±0.71	**1.46±0.47**
Hopper (ME)	s-vio (↓)	0.01±0.01	0.04±0.02	**0.00±0.00**	0.04±0.04	0.14±0.06	**0.00±0.00**	0.04±0.04	**0.00±0.00**
	time (↓)	**0.01±0.00**	1.13±0.01	1.66±0.03	0.15±0.00	0.15±0.00	1.03±0.02	0.16±0.00	0.05±0.00
	score (↑)	0.41±0.14	0.39±0.09	0.43±0.11	**0.46±0.04**	0.42±0.08	0.44±0.07	**0.46±0.05**	0.41±0.14
KUKA (Stacking)	s-vio (↓)	0.30±0.09	**0.00±0.00**	**0.00±0.00**	0.31±0.12	0.11±0.02	**0.00±0.00**	0.31±0.02	**0.00±0.00**
	time (↓)	**0.03±0.00**	38.14±0.65	37.21±0.08	21.25±0.19	22.02±0.05	68.75±1.65	25.01±0.21	0.10±0.00
	score (↑)	1.65±1.11	1.35±0.65	1.45±0.67	1.30±0.90	**1.75±0.94**	1.60±0.86	0.30±0.46	**1.75±0.83**

- **MeanFlow (MF)** (Geng et al., 2025) generates trajectories with a pretrained MeanFlow.
- **SafeFlowMatcher (SFM)** (Yang et al., 2026) generates a trajectory by a pretrained flow model in the prediction phase, and steers the trajectory toward a safe region by CBF in the correction phase.
- **SAD-Flower (SAD-F)** (Huang et al., 2026) trains a flow model and generates trajectories with a control-augmented framework to incorporate test-time safety constraints.
- **Diffuser** (Janner et al., 2022) is the pioneer work using a diffusion model to generate trajectories.
- **Classifier Guidance (CG)** (Dhariwal & Nichol, 2021) guides diffusion sampling using constraint gradients to steer generated trajectories toward safe regions during sampling.
- **SafeDiffuser (S-Diff.)** (Xiao et al., 2025) combines diffusion planning with CBF-based projection to enforce state constraints at test time throughout each denoise step.
- **CoBL** (Mizuta & Leung, 2024) incorporates physical constraints into diffusion sampling through shaping control barrier functions and control Lyapunov function as guidance.

5.2 Safety and Efficiency Across Simulation Tasks

For a fair evaluation, we compare all baselines under the same hardware and optimization software. As shown in Table 2, ASAP achieves the lowest computational cost among safety-guaranteed methods, planning at least 20× faster while matching their task performance. MF is marginally faster with single-step sampling but provides no safety guarantees.

In the Maze2d, ASAP achieves competitive performance while satisfying all safety constraints, showing that safety can be enforced without sacrificing performance when the constraint is compatible with the task, e.g., obstacles do not fully block the path. Other control-theoretic baselines also ensure safety, yet have higher planning costs due to repeated corrections during multi-step generation.

In the Hopper task, ASAP guarantees safety with the lowest computational cost while maintaining competitive performance, despite the height constraint restricting low-posture motions for faster locomotion. Other control-theoretic baselines also ensure safety but require longer planning time, while SFM still violates the constraint due to its slack variable. These results further show that ASAP efficiently enforces hard safety constraints without severely degrading task performance.

Similar trends are observed in the KUKA block-stacking task, where all methods require more generation steps to achieve strong performance. ASAP also benefits from slightly more steps, but still requires far fewer corrections than competing safety-guaranteed methods, remaining over 250× faster. This shows that ASAP can increase computation when beneficial for more complex tasks while retaining substantial efficiency. The further computational detail is reported in Appendix C.4.

5.3 Real-World Robotic Planning

We evaluate ASAP on a real-world service task where a Franka robot hands an ice cream to a customer through a shop window, as shown in Figure 2. A juicer is introduced as an obstacle in the test time to emulate constrained workspace geometry. Demonstrations are collected by teleoperation, with the robot picking the ice cream from a plate and delivering it to the human hand. At deploy-



Figure 2. Real-world handover. A robot passes an ice cream past a juicer that is absent from the demonstrations.

Table 3. Real-world results. ASAP ensures safety while planning over 200× faster than the safety-guaranteed baseline. Bold marks the best value.

Method	s-vio (↓)	time (↓)	score (↑)
MF	0.35±0.11	**0.01±0.01**	**1.00±0.01**
CG	0.27±0.12	2.18±0.23	0.99±0.01
S-Diff.	**0.00±0.00**	13.23±1.35	0.94±0.22
ASAP	**0.00±0.00**	0.04±0.01	**1.00±0.01**

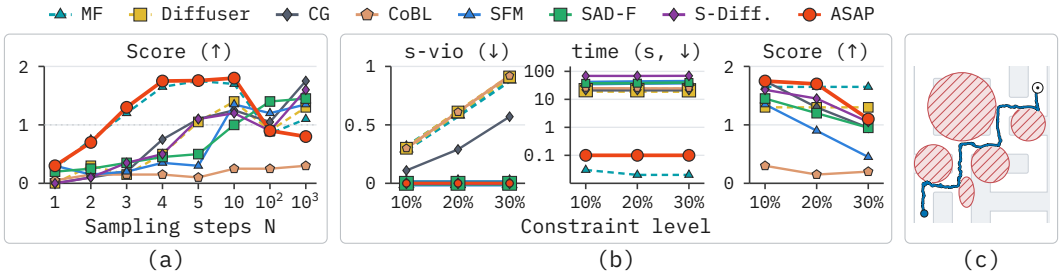

Figure 3. (a) Task score across sampling steps. For KUKA block stacking, the score of ASAP and MF passes 1 at $N = 3$ first, but only ASAP is safe. **(b) Tightening an unseen test-time constraint.** For KUKA block stacking, the feasible joint range progressively narrowed. ASAP holds safety at every level without retraining, at the lowest cost among safe methods; CoBL, Diffuser, CG, and MF violate more as it tightens. Overlapping curves are drawn thick-to-thin so each stays visible. **(c) Five simultaneous constraints.** Two unseen Maze2d obstacles become five and ASAP still plans collision-free at about 0.06 s.

ment, we assume a perception pipeline, e.g., a motion-capture system, provides accurate estimates of the ice-cream and target-hand positions.

Safety is measured by the minimum collision margin along the trajectory, normalized to $[0, 1]$. Task performance is based on final placement error, with a score of 1 for delivery within 1.0 cm of the target and higher scores for greater accuracy; the tolerance accounts for control error. We conduct 20 trials across two initial plate locations and three target hand positions, which define the ice-cream delivery targets, with trials approximately evenly distributed across these configurations. We compare against MF, CG, and S-Diff., which perform best in planning time, task score, and safety, respectively, on a similar manipulation task for the KUKA block-stacking simulation.

As shown in Table 3, ASAP matches the best safety performance of S-Diff. and the best task score of MF with only 0.04 s. MF needs less time but cannot guarantee safety under the unseen obstacle, while S-Diff. ensures safety at higher computational cost. These results demonstrate that ASAP enables efficient safe planning under unseen real-world constraints, with details in Appendix C.3.

5.4 How ASAP Enables Fast and Safe Planning

Performance of ASAP with Different Sampling Steps. We evaluate ASAP and all baselines across sampling steps N on the KUKA block-stacking task. As shown in Figure 3(a), ASAP and MF first exceed a task score of 1 at $N = 3$ and achieve the best performance for $N \leq 10$. This demonstrates that arbitrary-step safe generation enables strong performance with few steps while preserving safety. In contrast, most baselines require around 10 steps to exceed a score of 1, with some improving further at higher computational cost. MF degrades at larger N, likely due to training–inference mismatch in the pretrained MeanFlow model, which is optimized primarily for few-step generation. ASAP inherits this behavior from its MeanFlow backbone while remaining safe at every N.

Handling Multiple Safety Constraints. We evaluate ASAP under multiple safety constraints by increasing the number of obstacles in Maze2d from two to five. Multiple obstacles can be represented by a single SDF over the union of unsafe sets, or equivalently by selecting the nearest active obstacle SDF at differentiable points, with details in Appendix A. As shown in Figure 3(c), ASAP generates collision-free trajectories satisfying all constraints while maintaining a similar planning time of ap-

Table 4. QP vs. post-opt. Same MeanFlow model and constraints: the affine QP halves runtime with the same score and zero violation.

Maze2d	Post-opt.	ASAP
s-vio (↓)	0.000±0.000	0.000±0.000
time (↓)	0.033±0.015	**0.016±0.006**
score (↑)	1.456±0.467	1.456±0.467

Table 5. Barrier relaxation. On Maze2d with $N = 5$, tripling $\bar{h}$ from 1.0 to 3.0 in eq. (10) leaves metrics unaffected.

$\bar{h}$	s-vio (↓)	time (s) (↓)	score (↑)
1.0	0.00±0.00	0.25±0.01	1.46±0.47
2.0	0.00±0.00	0.25±0.01	1.46±0.47
3.0	0.00±0.00	0.25±0.01	1.46±0.47

Table 6. Adroit Relocate. In dexterous manipulation, ASAP is the fastest safe method, matching S-Diff. and SAD-F with zero violations.

Method	s-vio (↓)	time (↓)	score (↑)
MF	0.35±0.07	**0.02±0.00**	0.86±0.36
S-Diff.	**0.00±0.00**	0.59±0.01	**1.07±0.08**
SAD-F	**0.00±0.00**	1.54±0.15	1.05±0.23
ASAP	**0.00±0.00**	0.10±0.00	0.90±0.33

proximately 0.02 s on average. This low computational overhead is enabled by the lightweight QP formulation, demonstrating the flexibility of ASAP in handling multiple safety constraints.

Affine Barrier vs. Direct Post-Optimization. We compare ASAP with a naive post-optimization baseline to isolate the computational benefit of our QP formulation. Both use the same MeanFlow model and non-QP-capable solver for fairness. Post-optimization directly optimizes the trajectory under the original safety constraint, yielding a quadratically constrained quadratic program in Maze2d, whereas ASAP linearizes the barrier into an affine constraint and lightweight QP. As shown in Table 4, both achieve comparable task performance and satisfy the constraints, while ASAP roughly halves planning time, demonstrating the efficiency of the affine formulation.

5.5 How Reliable Is ASAP

Handling Stricter Test-Time Constraints. We evaluate ASAP under increasingly strict unseen constraints in the KUKA task by progressively tightening the feasible joint range for self-collision avoidance. As shown in Figure 3(b), ASAP satisfies all constraint levels without retraining, at the lowest planning cost among methods that stay safe. In contrast, CoBL, Diffuser, CG, and MF exhibit increasing violations as the constraints become stricter. Among methods that remain safe, ASAP also achieves the highest task score. The score decreases at the 30% constraint level because the tightened joint limits begin to restrict motions required for task completion.

Sensitivity Analysis of Hyperparameters. We evaluate the sensitivity of ASAP to $\bar{h}$, which controls the initial safety relaxation in eq. (10) with 5 sampling steps, by varying it from the default 1.0 to 3.0 while respecting the theoretical minimum. As shown in Table 5, ASAP remains safe across all tested values of $\bar{h}$ with similar task performance. These results show that ASAP is robust to $\bar{h}$ and does not require precise hyperparameter tuning.

Scalability to High-Dimensional Task. To evaluate scalability, we test ASAP on the D4RL Adroit Relocate task, a dexterous manipulation benchmark with 39-dimensional states and 30-dimensional actions. We impose collision-avoidance constraints based on the arm position. As shown in Table 6, ASAP is the fastest safety-guaranteed method and satisfies all constraints. MF plans faster but violates safety, while other control-theoretic baselines also ensure safety at over $5\times$ higher computational cost. These results show that ASAP scales effectively to high-dimensional tasks while preserving safety and efficiency. Further details are provided in Appendix C.2.

6 Conclusion

We proposed ASAP, a generative planner with formal safety guarantees for arbitrary-step generation. By integrating an affine barrier constraint into MeanFlow and adapting the safety correction to each sampling interval, ASAP enables fast trajectory generation while enforcing test-time safety constraints without retraining. Across navigation, locomotion, and manipulation tasks, ASAP satisfies every safety constraint while planning at least $20\times$ faster than safety-guaranteed baselines and maintaining competitive task performance; For real-world robot handover task, it remains safe and $200\times$ faster. Future work will extend the framework beyond current assumptions to dynamic consistency and richer constraint geometries, broadening its applicability to complex robotic systems.

AI USE STATEMENT

Generative AI tools, including large language models such as ChatGPT, were used to assist with language editing, rephrasing, LATEX formatting, and code implementation during manuscript preparation. They were not used to develop the research ideas, theoretical results, or core experimental methodology. All AI-assisted work was reviewed and revised by the authors, who take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

Code. Our implementation builds on the publicly available codebases of Flow Guidance (Feng et al., 2025) (`https://github.com/AI4Science-WestlakeU/flow_guidance`), SAD-Flower (Huang et al., 2026) (`https://github.com/TzuYuan107/sad-flower`), MeanFlow (Geng et al., 2025) (`https://github.com/Gsunshine/meanflow`), and MP1 (Sheng et al., 2026) (`https://github.com/LogSSim/MP1`). The code for ASAP will be released upon acceptance. Implementation and experimental details are provided in Section 5, with additional specifications in the Appendix.

Theory. All theoretical results presented in Section 4 are accompanied by the proof in Appendix B.

Datasets. We evaluate ASAP on publicly available datasets spanning navigation, locomotion, robotic manipulation, and dexterous manipulation. Dataset sources and preprocessing details are described in Section 5 and Appendix D.

Compute. All the training of each experiment are conducted using NVIDIA RTX 4090 GPUs. Detailed hardware configurations and computational resources are reported in Appendix G.

REFERENCES

Imoleayo Abel, Drew Steeves, Miroslav Krstić, and Mrdjan Janković. Prescribed-time safety design for strict-feedback nonlinear systems. *IEEE Transactions on Automatic Control*, 69(3):1464–1479, 2023.

Ayush Agrawal and Koushil Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *Robotics: Science and systems*, volume 13, pp. 1–10. Cambridge, MA, USA, 2017.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE transactions on automatic control*, 62(8): 3861–3876, 2016.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pp. 136–145. PMLR, 2017.

Azra Begzadić, Armin Lederer, Jorge Cortés, and Sylvia Herbert. Learning high-order cbfs using gaussian processes for systems in brunovskỳ canonical form. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pp. 2945–2950, 2025.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Ruiqi Feng, Chenglei Yu, Wenhao Deng, Peiyan Hu, and Tailin Wu. On the guidance of flow matching. In *International Conference on Machine Learning*, pp. 16993–17029, 2025.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Riku Funada, Koju Nishimoto, Tatsuya Ibuki, and Mitsuji Sampei. Collision avoidance for ellipsoidal rigid bodies with control barrier functions designed from rotating supporting hyperplanes. *IEEE Transactions on Control Systems Technology*, 33(1):148–164, 2024.

Yuan Gao, Jian Huang, Yuling Jiao, and Shurong Zheng. Convergence of continuous normalizing flows for learning probability distributions. *arXiv preprint arXiv:2404.00551*, 2024.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pp. 440–448, 2020.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2025.

David Gilbarg, Neil S Trudinger, David Gilbarg, and NS Trudinger. *Elliptic partial differential equations of second order*, volume 224. Springer, 1977.

Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE control systems letters*, 1(2):310–315, 2017.

Jaskaran Grover, Changliu Liu, and Katia Sycara. The before, during, and after of multi-robot deadlock. *The International Journal of Robotics Research*, 42(6):317–336, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

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

Songqiao Hu, Zeyi Liu, Shuang Liu, Jun Cen, Zihan Meng, Shihefeng Wang, Xiang Li, and Xiao He. Vlsa: Vision-language-action models with plug-and-play safety constraint layer. *arXiv preprint arXiv:2512.11891*, 2025.

Tzu-Yuan Huang, Armin Lederer, Nicolas Hoischen, Jan Brüdigam, Xuehua Xiao, Stefan Sosnowski, and Sandra Hirche. Toward near-globally optimal nonlinear model predictive control via diffusion models. *arXiv preprint arXiv:2412.08278*, 2024a.

Tzu-Yuan Huang, Sihua Zhang, Xiaobing Dai, Alexandre Capone, Velimir Todorovski, Stefan Sosnowski, and Sandra Hirche. Learning-based prescribed-time safety for control of unknown systems with control barrier functions. *IEEE Control Systems Letters*, 8:1817–1822, 2024b.

Tzu-Yuan Huang, Armin Lederer, Dai-Jie Wu, Xiaobing Dai, Sihua Zhang, Hsiu-Chin Lin, Shao-Hua Sun, Stefan Sosnowski, and Sandra Hirche. Sad-flower: Flow matching for safe, admissible, and dynamically consistent planning. *International Conference on Machine Learning*, 2026.

Wei Hung, Shao-Hua Sun, and Ping-Chun Hsieh. Efficient action-constrained reinforcement learning via acceptance-rejection method and augmented mdps. In *International Conference on Learning Representations*, volume 2025, pp. 20573–20595, 2025.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *International Conference on Machine Learning*, 2022.

Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pp. 4569–4574. IEEE, 2011.

Kota Kondo, Andrea Tagliabue, Xiaoyi Cai, Claudius Tewari, Olivia Garcia, Marcos Espitia-Alvarez, and Jonathan P How. Cgd: Constraint-guided diffusion policies for uav trajectory planning. *arXiv preprint arXiv:2405.01758*, 2024.

Anjian Li, Zihan Ding, Adji Bousso Dieng, and Ryne Beeson. Diffusolve: Diffusion-based solver for non-convex trajectory optimization. *arXiv preprint arXiv:2403.05571*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Kehan Long, Cheng Qian, Jorge Cortés, and Nikolay Atanasov. Learning barrier functions with memory for robust safe navigation. *IEEE Robotics and Automation Letters*, 6(3):4931–4938, 2021.

Kehan Long, Yinzhuang Yi, Zhirui Dai, Sylvia Herbert, Jorge Cortés, and Nikolay Atanasov. Sensor-based distributionally robust control for safe robot navigation in dynamic environments. *The International Journal of Robotics Research*, 45(2):328–351, 2026.

François Mazé and Faez Ahmed. Diffusion models beat gans on topology optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 9108–9116, 2023.

Gonzalo Mena, Arun Kumar Kuchibhotla, and Larry Wasserman. Statistical properties of rectified flow. *arXiv preprint arXiv:2511.03193*, 2025.

Kazuki Mizuta and Karen Leung. Cobl-diffusion: Diffusion-based conditional robot planning in dynamic environments using control barrier and lyapunov functions. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13801–13808. IEEE, 2024.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 165–174. IEEE, 2019.

Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Ralf Römer, Alexander von Rohr, and Angela P Schoellig. Diffusion predictive control with constraints. *arXiv preprint arXiv:2412.09342*, 2024.

Juyi Sheng, Ziyi Wang, Peiming Li, and Mengyuan Liu. Mp1: Meanflow tames policy learning in 1-step for robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18532–18539, 2026.

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.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

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.

Junli Wang, Yinan Zheng, Xueyi Liu, Zebin Xing, Pengfei Li, Kun Ma, Hangjun Ye, Guang Chen, Guang Li, Long Chen, et al. Meanfuser: Fast one-step multi-modal trajectory generation and adaptive reconstruction via meanflow for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17884–17893, 2026.

Wei Xiao, Tsun-Hsuan Wang, Chuang Gan, Ramin Hasani, Mathias Lechner, and Daniela Rus. Safediffuser: Safe planning with diffusion probabilistic models. In *International Conference on Learning Representations*, 2025.

Jeongyong Yang, Seunghwan Jang, and SooJean Han. Safeflowmatcher: Safe and fast planning using flow matching with control barrier functions. In *International Conference on Learning Representations*, volume 2026, pp. 54058–54087, 2026.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

Appendix

Table of Contents

A Signed Distance Functions **14**

B Proofs of Theoretical Results **15**
- B.1 Proof of lemma 4.1 . . . 15
- B.2 Proof of theorem 4.1 . . . 15
- B.3 Proof of corollary 4.1 . . . 16
- B.4 Proof of theorem 4.2 . . . 16
- B.5 Feasibility of Arbitrary-Step Affine Barrier Constraints . . . 17
- B.6 Extension to General Barrier Functions . . . 17

C Additional Evaluation **20**
- C.1 Performance Across Navigation, Locomotion, and Manipulation . . . 20
- C.2 Performance on High-Dimensional Tasks . . . 21
- C.3 Real-World Robotic Planning Results . . . 21
- C.4 Computational Scalability of the QP. . . . 22
- C.5 Practical Evaluation of Conservatism . . . 22

D Environment Details **22**

E Test-Time Safety Constraints **22**

F Training Details **23**

G Computational Resources **24**

A Signed Distance Functions

Given a constraint set $\mathbb{S}$ with boundary $\partial\mathbb{S}$, a signed distance function (SDF) measures the shortest distance to the boundary, while its sign indicates whether the constraint is satisfied:

$$h(\boldsymbol{\tau}_{t_i}) = \begin{cases} \min_{\boldsymbol{p}\in\partial\mathbb{S}} ||\boldsymbol{\tau}_{t_i} - \boldsymbol{p}|| & \text{if } \boldsymbol{\tau}_{t_i} \in \mathbb{S} \\ -\min_{\boldsymbol{p}\in\partial\mathbb{S}} ||\boldsymbol{\tau}_{t_i} - \boldsymbol{p}|| & \text{else} \end{cases} \tag{15}$$

Accordingly, $h(\boldsymbol{x}) \geq 0$ indicates constraint satisfaction. In trajectory planning, this SDF can be evaluated at each trajectory state $\boldsymbol{\tau}_{t_i}(k)$ to obtain the state-wise barrier $h(\boldsymbol{\tau}_{t_i}(k))$.

Multiple constraints as a single SDF. Multiple obstacle constraints can be represented by a single SDF by first combining the unsafe sets. Let $\{\mathbb{O}_j\}_{j=1}^{M}$ denote M unsafe obstacle sets. The total unsafe region is

$$\mathbb{O} = \bigcup_{j=1}^{M} \mathbb{O}_j, \tag{16}$$

and the corresponding safe set is

$$\mathbb{S} = \mathbb{R}^d \setminus \mathbb{O}. \tag{17}$$

The SDF in eq. (15) can then be defined directly with respect to this combined safe set $\mathbb{S}$. Thus, even with multiple obstacles, safety can still be expressed through a single scalar condition

$$h(\boldsymbol{x}) \geq 0. \tag{18}$$

For common obstacle-avoidance constraints, this combined SDF has a simple form. If each obstacle $\mathbb{O}_j$ has an individual SDF $h_j(\boldsymbol{x})$ with positive value outside the obstacle, then for non-overlapping obstacles the combined SDF is

$$h(\boldsymbol{x}) = \min_{j=1,\dots,M} h_j(\boldsymbol{x}). \tag{19}$$

Equivalently, this follows from the distance to a union:

$$\operatorname{dist}\left(\boldsymbol{x}, \bigcup_{j=1}^{M} \mathbb{O}_j\right) = \min_{j=1,\dots,M} \operatorname{dist}(\boldsymbol{x}, \mathbb{O}_j). \tag{20}$$

Therefore, the combined SDF automatically selects the closest active obstacle. At a differentiable point where the nearest obstacle is unique, say

$$j^\star = \arg\min_j h_j(\boldsymbol{x}), \tag{21}$$

we have

$$h(\boldsymbol{x}) = h_{j^\star}(\boldsymbol{x}), \qquad \nabla h(\boldsymbol{x}) = \nabla h_{j^\star}(\boldsymbol{x}). \tag{22}$$

Thus, the SDF gradient points toward the nearest active safety boundary, which is the direction used by the minimum-norm affine barrier correction in our method.

At points where multiple obstacles are equally close, the minimum composition may be non-differentiable. Such points correspond to ties between nearest boundaries. In practice, they can be handled by generalized gradients, smoothing, or active-boundary selection, as discussed in Section 4.2. These cases do not change the sign condition $h(\boldsymbol{x}) \geq 0$, and the differentiable case covers almost all regular points.

Examples. For circular obstacles $\mathbb{O}_j = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{c}_j\| \leq r_j\}$, the individual SDF is

$$h_j(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{c}_j\| - r_j. \tag{23}$$

The combined SDF for avoiding all obstacles is

$$h(\boldsymbol{x}) = \min_{j=1,\dots,M} \left(\|\boldsymbol{x} - \boldsymbol{c}_j\| - r_j\right). \tag{24}$$

Then $h(\boldsymbol{x}) \geq 0$ holds if and only if $\boldsymbol{x}$ is outside every obstacle.

Equivalent barrier functions that preserve the same zero level set and sign can also be used. For instance, collision avoidance outside a circle can be represented by $h(x, y) = (x - x_0)^2 + (y - y_0)^2 - r^2 \geq 0$, and more generally, an ellipsoidal constraint can be expressed through a quadratic barrier of the form $h(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{c})^\top Q(\boldsymbol{x} - \boldsymbol{c}) - 1 \geq 0$.

B Proofs of Theoretical Results

B.1 Proof of lemma 4.1

Proof of Lemma 4.1. The time-varying CBF in eq. (5) is equilavant to $\dot{B} + B \geq 0$, where $B := B_t$ and $B_t = h(\boldsymbol{\tau}_t) - \rho_t$. The initial barrier function $B_0 \geq 0$ due to our design of ρ_0, based on the scheduling design of ρ_t, by solving the QP in eq. (5), , it will remain in the positive region for the full time $B_t \geq 0, t \in [0, 1]$ with control barrier function theory (Ames et al., 2016). Since $\rho_1 = 0$, the barrier function $B_1 = h(\boldsymbol{\tau}_1) \geq 0$, which implies the safe trajectory generation. This formulation is similar to the prescribed-time control barrier function (Abel et al., 2023; Huang et al., 2024b), which designs a time-varying CBF to guarantee safety constraints at a predefined time. □

B.2 Proof of theorem 4.1

Proof of Theorem 4.1. If $h(\hat{\boldsymbol{\tau}}_{t_i}) \geq \rho_{t_{i+1}}$, the minimum-norm QP returns $\boldsymbol{u}_{t_i} = \mathbf{0}$, so the next-step condition holds. Otherwise, $h(\hat{\boldsymbol{\tau}}_{t_i}) < \rho_{t_{i+1}} \leq 0$. The exact SDF has a unit gradient at this unsafe nominal point, and the minimum-norm solution is (Ames et al., 2016)

$$\boldsymbol{u}_{t_i} \Delta t = \frac{\left(\rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i})\right)}{\|\nabla h(\hat{\boldsymbol{\tau}}_{t_i})\|^2} \nabla h(\hat{\boldsymbol{\tau}}_{t_i}) = \left(\rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i})\right) \nabla h(\hat{\boldsymbol{\tau}}_{t_i}). \tag{25}$$

Let $\boldsymbol{p} \in \partial\mathbb{S}$ be a nearest safe point to $\hat{\boldsymbol{\tau}}_{t_i}$. Differentiability of the exact SDF implies $\boldsymbol{p} - \hat{\boldsymbol{\tau}}_{t_i} = -h(\hat{\boldsymbol{\tau}}_{t_i})\nabla h(\hat{\boldsymbol{\tau}}_{t_i})$. For a displacement of length $0 \le s \le -h(\hat{\boldsymbol{\tau}}_{t_i})$ along this gradient, the triangle inequality gives the lower bound below, while the candidate $\boldsymbol{p} \in \mathbb{S}$ gives the upper bound:

$$\begin{aligned}
&\operatorname{dist}\big(\hat{\boldsymbol{\tau}}_{t_i} + s\nabla h(\hat{\boldsymbol{\tau}}_{t_i}), \mathbb{S}\big) \ge \operatorname{dist}(\hat{\boldsymbol{\tau}}_{t_i}, \mathbb{S}) - s = -h(\hat{\boldsymbol{\tau}}_{t_i}) - s, \\
&\operatorname{dist}\big(\hat{\boldsymbol{\tau}}_{t_i} + s\nabla h(\hat{\boldsymbol{\tau}}_{t_i}), \mathbb{S}\big) \le \big\|\hat{\boldsymbol{\tau}}_{t_i} + s\nabla h(\hat{\boldsymbol{\tau}}_{t_i}) - \boldsymbol{p}\big\| = -h(\hat{\boldsymbol{\tau}}_{t_i}) - s.
\end{aligned}$$

The two bounds coincide. The corrected point remains outside $\mathbb{S}$ until it reaches $\boldsymbol{p}$, so its signed distance satisfies

$$h\big(\hat{\boldsymbol{\tau}}_{t_i} + s\nabla h(\hat{\boldsymbol{\tau}}_{t_i})\big) = h(\hat{\boldsymbol{\tau}}_{t_i}) + s, \qquad 0 \le s \le -h(\hat{\boldsymbol{\tau}}_{t_i}). \tag{26}$$

Since $0 < \rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i}) \le -h(\hat{\boldsymbol{\tau}}_{t_i})$, the optimal correction stays on this segment that $s^* = \rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i})$. With $\boldsymbol{\tau}_{t_{i+1}} = \hat{\boldsymbol{\tau}}_i + s^*\nabla h(\hat{\boldsymbol{\tau}}_{t_i})$ induced by eq. (25) and our update rule in eq. (9), we can obtain the following equation

$$h(\boldsymbol{\tau}_{t_{i+1}}) = h\big(\hat{\boldsymbol{\tau}}_{t_i} + s^*\nabla h(\hat{\boldsymbol{\tau}}_{t_i})\big) = h(\hat{\boldsymbol{\tau}}_{t_i}) + s^* = \rho_{t_{i+1}}, \qquad i = 0, \dots, N-1. \tag{27}$$

At the final step, $\rho_{t_N} = 0$: an unsafe nominal state is corrected exactly to the nearest safe boundary, while an already safe nominal state is unchanged. Applying this argument to every trajectory state proves terminal safety for any $N \ge 1$. The terminal sample is $\boldsymbol{\tau}_{t_N}$ in discrete-step notation, corresponding to $\boldsymbol{\tau}_1$ at generative time $t = 1$. □

The assumption of Theorem 4.1 requires the feasibility during QP solving. Even though it is a well-known topic of the standard CBF-based method, it is typically not an issue for our case due to the fully actuated, integrator-like generation dynamics in eq. (9). We will discuss the details of feasibility in Theorem B.1.

B.3 Proof of corollary 4.1

Proof of Corollary 4.1. Holding the affine barrier function eq. (12), we can directly obtain the following inequality by setting $N = 1$ for Theorem 4.1:

$$h(\boldsymbol{\tau}_1) = h(\hat{\boldsymbol{\tau}}_0) + \nabla h(\hat{\boldsymbol{\tau}}_0)^\top(\boldsymbol{u}_0\Delta t) \ge \rho_1, \quad \Delta t = 1 \tag{28}$$

Since $\rho_1 = 0$, we can prove that the trajectory $\boldsymbol{\tau}_1$ is safe at $t = 1$ in one step generation. □

The Corollary 4.1 can be viewed as a special case of Theorem 4.1. When $N = 1$, it shows that terminal safety can still be guaranteed with only a single generative update and one safety correction, thereby enabling one-step safe trajectory generation.

B.4 Proof of theorem 4.2

Proof of Theorem 4.2. Consider an arbitrary sampling step i and denote

$$\boldsymbol{g}_{t_i} := \nabla h(\hat{\boldsymbol{\tau}}_{t_i}), \qquad b_{t_i} := \rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i}).$$

The affine barrier constraint in eq. (12) becomes

$$\boldsymbol{g}_{t_i}^\top(\boldsymbol{u}_{t_i}\Delta t) \ge b_{t_i}.$$

If $b_{t_i} \le 0$, the nominal sample already satisfies the affine barrier constraint. Hence, the minimum-norm QP in eq. (13) returns $\boldsymbol{u}_{t_i} = \boldsymbol{0}$, such that $\boldsymbol{\tau}_{t_{i+1}} = \hat{\boldsymbol{\tau}}_{t_i}$. Therefore, the barrier value equals the true next-step barrier value and $C_i = 0$. If $b_{t_i} > 0$, the barrier constraint is active. The minimum-norm solution aligns with $\boldsymbol{g}_{t_i}$, since any component orthogonal to $\boldsymbol{g}_{t_i}$ increases the objective without affecting the constraint. Enforcing the active constraint with equality gives

$$\boldsymbol{u}_{t_i}\Delta t = \frac{b_{t_i}}{\|\boldsymbol{g}_{t_i}\|^2}\boldsymbol{g}_{t_i}.$$

Since h is an exact differentiable SDF and $b_{t_i} > 0$ implies $h(\hat{\boldsymbol{\tau}}_{t_i}) < \rho_{t_{i+1}} \le 0$, $\hat{\boldsymbol{\tau}}_{t_i}$ lies outside the safe set and $||\boldsymbol{g}_{t_i}|| = 1$. Moreover, $\boldsymbol{g}_{t_i}$ points along the shortest direction toward the nearest safe boundary. Let $s \ge 0$ denote the traveled distance along this unit direction. The SDF therefore satisfies

$$h(\hat{\boldsymbol{\tau}}_{t_i} + s\boldsymbol{g}_{t_i}) = h(\hat{\boldsymbol{\tau}}_{t_i}) + s, \qquad 0 \le s \le -h(\hat{\boldsymbol{\tau}}_{t_i}). \tag{29}$$

Because $b_{t_i} = \rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i}) \leq -h(\hat{\boldsymbol{\tau}}_{t_i})$, the QP correction remains within this interval. Using $||\boldsymbol{g}_{t_i}|| = 1$, we have $\boldsymbol{u}_{t_i}\Delta t = b_{t_i}\boldsymbol{g}_{t_i}$. Following the update rule eq. (9) that $\boldsymbol{\tau}_{t_{i+1}} = \hat{\boldsymbol{\tau}}_{t_i} + \boldsymbol{u}_{t_i}\Delta t$, the exact value of SDF at next step $h(\boldsymbol{\tau}_{t_{i+1}})$ is $h(\boldsymbol{\tau}_{t_{i+1}}) = h(\hat{\boldsymbol{\tau}}_{t_i} + \boldsymbol{u}_{t_i}\Delta t) = h(\hat{\boldsymbol{\tau}}_{t_i} + b_{t_i}\boldsymbol{g}_{t_i})$. Based on the eq. (29), we can derive this exact value as follows

$$h(\boldsymbol{\tau}_{t_{i+1}}) = h(\hat{\boldsymbol{\tau}}_{t_i} + b_{t_i}\boldsymbol{g}_{t_i}) = h(\hat{\boldsymbol{\tau}}_{t_i}) + b_{t_i} = \rho_{t_{i+1}}. \tag{30}$$

The affine formulation from eq. (12) yields the same value:

$$\hat{h}(\hat{\boldsymbol{\tau}}_{t_i}) = h(\hat{\boldsymbol{\tau}}_{t_i}) + \boldsymbol{g}_{t_i}^\top(\boldsymbol{u}_{t_i}\Delta t) = h(\hat{\boldsymbol{\tau}}_{t_i}) + \boldsymbol{g}_{t_i}^\top\boldsymbol{g}_i b_{t_i} = h(\hat{\boldsymbol{\tau}}_{t_i}) + b_{t_i} = \rho_{t_{i+1}}. \tag{31}$$

Thus, its barrier value and the true next-step barrier value coincide, i.e., $h(\boldsymbol{\tau}_{t_{i+1}}) = \hat{h}(\hat{\boldsymbol{\tau}}_{t_i})$, which gives $C_i = 0$ by eq. (14). Since this holds for every $i = 0, \ldots, N-1$, $\bar{C} = 0$ for arbitrary N. □

B.5 Feasibility of Arbitrary-Step Affine Barrier Constraints

Theorem B.1. *Suppose $h(\cdot)$ is differentiable for each $\hat{\boldsymbol{\tau}}_{t_i}$,, then, for every generation step i, there exists a correction $\boldsymbol{u}_{t_i}$ satisfying the affine barrier constraint eq.* (12)*.*

Proof. The affine barrier constraint can be written as

$$\nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top(\boldsymbol{u}_{t_i}\Delta t) \geq \rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i}). \tag{32}$$

If $h(\hat{\boldsymbol{\tau}}_{t_i}) \geq \rho_{t_{i+1}}$, the constraint is already satisfied by $\boldsymbol{u}_{t_i} = \mathbf{0}$. Otherwise, a feasible correction can always be chosen along the barrier gradient, e.g.,

$$\boldsymbol{u}_{t_i} = \frac{\rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i})}{\Delta t\,\|\nabla h(\hat{\boldsymbol{\tau}}_{t_i})\|^2}\nabla h(\hat{\boldsymbol{\tau}}_{t_i}). \tag{33}$$

This choice satisfies the affine barrier constraint with equality, establishing feasibility at every step. □

For an exact signed distance function (SDF), differentiability implies $\|\nabla h\| = 1$ wherever the SDF is differentiable, so the non-vanishing-gradient condition in theorem B.1 is naturally satisfied. SDFs may nevertheless be non-differentiable at certain points, although they are differentiable almost everywhere under standard regularity conditions on the underlying set (Gilbarg et al., 1977). In practice, such non-smoothness can be handled through smooth approximations (Begzadić et al., 2025) or by decomposing a complex constraint set into multiple subsets and constructing a separate barrier for each subset.

For example, the commonly used box constraint $\|\boldsymbol{\tau}_{t_i}(k)\|_\infty \leq \bar{\tau}$ can be decomposed into individual upper and lower bounds, yielding affine barriers of the form $h_k(\boldsymbol{\tau}_{t_i}) = \bar{\tau} \pm \boldsymbol{\tau}_{t_i}(k)$. These barriers are differentiable with constant nonzero gradients. Consequently, the regularity condition required by theorem B.1 is readily satisfied for many practical constraint constructions used in CBF-based planning (Long et al., 2021).

B.6 Extension to General Barrier Functions

In the main paper, we instantiate the barrier function h as an exact signed distance function (SDF), which provides a natural representation of geometric safety constraints and informative gradients toward the safe set (Long et al., 2021; Park et al., 2019). Importantly, its geometric properties enable the safety guarantee in Theorem 4.1 without requiring convexity of h. Nevertheless, ASAP is not restricted to exact SDFs. We next show that arbitrary-step safety can also be guaranteed for a general differentiable barrier function under a convexity assumption.

Theorem B.2 (Arbitrary-Step Safety with Convex Barriers). *If h is differentiable and convex, the control input $\boldsymbol{u}_{t_i}$ given by eq.* (13) *ensures that the solution $\boldsymbol{\tau}_{t_N}$ of eq.* (9) *satisfies the safety constraint $h(\boldsymbol{\tau}_{t_N}) \geq 0$ for any initial condition $\boldsymbol{\tau}_0$,*

Proof. By convexity of h, its first-order approximation provides a global lower bound:

$$h(\boldsymbol{\tau}_{t_{i+1}}) = h(\hat{\boldsymbol{\tau}}_{t_i} + \boldsymbol{u}_{t_i}\Delta t) \geq h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top(\boldsymbol{u}_{t_i}\Delta t) \geq \rho_{t_{i+1}}, \tag{34}$$

where the second inequality follows from eq. (12). Hence, $h(\boldsymbol{\tau}_{t_{i+1}}) \geq \rho_{t_{i+1}}$ for every $i = 0, \ldots, N-1$. Since $\rho_{t_N} = 0$, the terminal sample $\boldsymbol{\tau}_{t_N}$, corresponding to $\boldsymbol{\tau}_1$ at generative time $t = 1$, satisfies $h(\boldsymbol{\tau}_1) \geq 0$. □

Remark B.1. *The convexity assumption is imposed on the certificate function $h(\cdot)$, rather than on its safe superlevel set $\mathcal{C} = \{\boldsymbol{\tau} : h(\boldsymbol{\tau}) \geq 0\}$, and therefore does not restrict the safe region to be convex. In particular, commonly used distance-based collision-avoidance barriers, such as $h(p) = ||p - c||^2 - r^2$, for spherical obstacles (Grover et al., 2023), are convex, while their corresponding collision-free sets are nonconvex. The same property extends naturally to ellipsoidal obstacles through quadratic barriers $h(p) = (p - c)^T Q(p - c) - 1, Q \succ 0$, which are closely related to ellipsoidal CBF constructions widely used in robotic collision avoidance (Funada et al., 2024; Hu et al., 2025). Hence, the convexity assumption accommodates a broad class of practically relevant geometric safety constraints.*

Relaxation to Nonconvex Barrier Functions. The convexity requirement can be further relaxed when the negative curvature of h is bounded. Specifically, assume that h is twice differentiable and

$$\nabla^2 h(\boldsymbol{\tau}) \succeq -\mu \boldsymbol{I}, \qquad \mu \in \mathbb{R}_+, \tag{35}$$

over the relevant domain, and an upper bound of control $|\boldsymbol{u}_{t_i}| \leq \bar{U}$, where we discuss at the end of this section. We obtain the following result.

Theorem B.3 (**Safety with Bounded Nonconvexity**). *Under the assumptions above, the solution $\boldsymbol{\tau}_{t_N}$ of eq.* (9) *is safe if the control input $\boldsymbol{u}_{t_i}$ satisfies*

$$h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i} \Delta t) \geq \rho_{t_{i+1}} + \frac{\mu \bar{U}^2}{2N^2}. \tag{36}$$

Proof. Let $\boldsymbol{d}_{t_i} = \boldsymbol{u}_{t_i} \Delta t$. By Taylor's theorem, for some $\xi \in [0, 1]$,

$$h(\boldsymbol{\tau}_{t_{i+1}}) = h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top \boldsymbol{d}_{t_i} + \frac{1}{2} \boldsymbol{d}_{t_i}^\top \nabla^2 h(\hat{\boldsymbol{\tau}}_{t_i} + \xi \boldsymbol{d}_{t_i}) \boldsymbol{d}_{t_i}. \tag{37}$$

The bounded-curvature assumption implies

$$\frac{1}{2} \boldsymbol{d}_{t_i}^\top \nabla^2 h(\hat{\boldsymbol{\tau}}_{t_i} + \xi \boldsymbol{d}_{t_i}) \boldsymbol{d}_{t_i} \geq -\frac{\mu}{2} ||\boldsymbol{d}_{t_i}||^2. \tag{38}$$

Since $|\boldsymbol{u}_{t_i}| \leq \bar{U}$ (see the discussion at the end of this section) and $\Delta t = 1/N$, the right term in eq. (38) follows $-\frac{\mu}{2} ||\boldsymbol{d}_{t_i}||^2 \geq -\frac{\mu \bar{U}^2}{2N^2}$ and we can rewrite eq. (37) as follow

$$h(\boldsymbol{\tau}_{t_{i+1}}) \geq h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i} \Delta t) - \frac{\mu \bar{U}^2}{2N^2}. \tag{39}$$

If the robust barrier function in eq. (36) is satisfied, inducing $h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i} \Delta t) - \frac{\mu \bar{U}^2}{2N^2} \geq \rho_{t_{i+1}}$, we can prove that $\boldsymbol{\tau}_{t_N}$ is safe since $h(\boldsymbol{\tau}_{t_{i+1}}) \geq h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i} \Delta t) - \frac{\mu \bar{U}^2}{2N^2} \geq \rho_{t_{i+1}}$. Therefore, satisfying eq. (36) yields $h(\boldsymbol{\tau}_{t_{i+1}}) \geq \rho_{t_{i+1}}$ for every step $i = 0, \cdots, N - 1$. Since $\rho_{t_N} = 0$, terminal safety follows. □

The additional term $\frac{\mu \bar{U}^2}{2N^2}$ acts as a robust margin accounting for the bounded negative curvature of h. With Theorem B.3, we can apply our ASAP framework with a robust barrier function as defined in eq. (36) and keep an efficient QP formulation even if $h(\cdot)$ is non-convex. For $\mu = 0$, h is convex and eq. (36) recovers the original affine barrier constraint in eq. (12).

For the conservatism analysis of the general CBF, we derive it as follows.

Theorem B.4 (**Arbitrary-Step Conservatism Bound for general CBF**). *If $h(\cdot)$ is L-smooth, i.e., $||\nabla^2 h(\boldsymbol{\tau})||_2 \leq L$ and convex $||\nabla^2 h(\cdot)|| \geq 0$, where L is the Lipschitz constant of ∇h, the conservatism $\bar{C}$ induced by eq.* (13) *is bounded and decreases with the number of sampling steps N as*

$$\bar{C} \leq \frac{L \bar{U}^2}{2N}, \quad \textit{where } \bar{C} = O(1/N) \tag{40}$$

This theorem shows that the conservatism induced by the finite-step affine approximation remains bounded even for small N. As $N \to \infty$ and $\Delta t \to 0$, the approximation-induced conservatism vanishes, recovering the continuous-time barrier evaluation. Thus, our analysis characterizes the trade-off between sampling efficiency and conservatism through the choice of N.

Remark B.2. *The L-smoothness assumption is mild for SDF-based constraints. Although exact SDFs can be non-smooth at certain points, they are smooth with Lipschitz-continuous gradients over regular regions of the planning domain. Moreover, SDFs can be smoothly approximated (Begzadić et al., 2025) or constructed over simpler decomposed constraint sets to obtain L-smooth barriers. Thus, L-smoothness can be readily satisfied for practical SDF-based safety constraints.*

Proof of Theorem B.4. The constraint residual is defined for each step as

$$C_i = h(\boldsymbol{\tau}_{t_{i+1}}) - h(\hat{\boldsymbol{\tau}}_{t_i}) - \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i}\Delta t), \quad \forall i = 0, \dots, N-1, \tag{41}$$

where the first-order approximation is $\hat{h}(\hat{\boldsymbol{\tau}}_{t_i}) = h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i}\Delta t)$. By Taylor's theorem, we can obtain the exact barrier value of the next step $h(\boldsymbol{\tau}_{t_{i+1}})$ as

$$h(\boldsymbol{\tau}_{t_{i+1}}) = h(\hat{\boldsymbol{\tau}}_{t_i}) + \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i}\Delta t) + \frac{1}{2}(\boldsymbol{u}_{t_i}\Delta t)^\top \nabla^2 h(\hat{\boldsymbol{\tau}}_{t_i} + \xi \boldsymbol{u}_{t_i}\Delta t)(\boldsymbol{u}_{t_i}\Delta t), \tag{42}$$

where the second-order prediction has an upper bound if $h(\cdot)$ is L-smooth

$$(\boldsymbol{u}_{t_i}\Delta t)^\top \nabla^2 h(\hat{\boldsymbol{\tau}}_{t_i} + \xi \boldsymbol{u}_{t_i}\Delta t)(\boldsymbol{u}_{t_i}\Delta t) \leq (\boldsymbol{u}_{t_i}\Delta t)^\top L I (\boldsymbol{u}_{t_i}\Delta t) = L||\boldsymbol{u}_{t_i}\Delta t||^2. \tag{43}$$

By substituting eq. (43) to eq. (42), we can obtain the inequality as follows:

$$h(\boldsymbol{\tau}_{t_{i+1}}) - h(\hat{\boldsymbol{\tau}}_{t_i}) - \nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top (\boldsymbol{u}_{t_i}\Delta t) \leq \frac{L}{2}||\boldsymbol{u}_{t_i}\Delta t||^2. \tag{44}$$

As a result, we can derive the upper bound of the constraint residual at each step

$$C_i \leq \frac{L}{2}||\boldsymbol{u}_{t_i}\Delta t||^2 \tag{45}$$

and the total conservatism is shown as follows

$$\bar{C} = \sum_{i=0}^{N-1} C_i \leq \frac{L}{2} \sum_{i=0}^{N-1} \|\boldsymbol{u}_{t_i}\Delta t\|^2. \tag{46}$$

Since the $||\boldsymbol{u}_{t_i}|| \leq \bar{U}$ and $\Delta t = \frac{1}{N}$, we can prove that $\bar{C} \leq \frac{L\bar{U}^2}{2N}$. □

How to estimate the value of $\bar{U}$? For our ASAP framework, we calculate $\boldsymbol{u}_{t_i}$ by eq. (13) to minimally perturb the MeanFlow update with following constraint

$$\nabla h(\hat{\boldsymbol{\tau}}_{t_i})^\top \boldsymbol{u}_{t_i}\Delta t \geq d_i, \tag{47}$$

where the d_i is the scheduled deficit $d_i = \rho_{t_{i+1}} - h(\hat{\boldsymbol{\tau}}_{t_i}) = \rho_{t_{i+1}} - \rho_{t_i} - (h(\hat{\boldsymbol{\tau}}_{t_i}) - \rho_{t_i})$. Based on the time-varying scheduled term ρ_{t_i} defined in eq. (10), we can derive a weighting term w_i that schedule the constraint level with the range $[-\bar{h}, 0]$ evenly as follow

$$w_i = \frac{(\rho_{t_{i+1}} - \rho_{t_i})}{\bar{h}} = \frac{1}{N}. \tag{48}$$

For the deviation between $h(\hat{\boldsymbol{\tau}}_{t_i})$ and ρ_{t_i}, we can obtain the following inequality if $h(\cdot)$ is Lipschitz, which is a mild assumption in real-world applications (Grover et al., 2023)

$$\|h(\hat{\boldsymbol{\tau}}_{t_i}) - h(\boldsymbol{\tau}_{t_i})\| \leq G||\boldsymbol{\mu}^{\boldsymbol{\theta}}_{t_i}||\Delta t, \tag{49}$$

where the $G \in \mathbb{R}_+$ is the Lipschitz constant of the $h(\cdot)$ and $\Delta t = \frac{1}{N}$. By reformulating eq. (49) as $h(\hat{\boldsymbol{\tau}}_{t_i}) - h(\boldsymbol{\tau}_{t_i}) \geq -G||\boldsymbol{\mu}^{\boldsymbol{\theta}}_{t_i}||\Delta t$ by Absolute Value Inequalities, we can derive

$$h(\hat{\boldsymbol{\tau}}_{t_i}) \geq h(\boldsymbol{\tau}_{t_i}) - \frac{G||\boldsymbol{\mu}^{\boldsymbol{\theta}}_{t_i}||}{N} \geq \rho_{t_i} - \frac{GV}{N}, \tag{50}$$

if MeanFlow is bounded $\|\boldsymbol{\mu}_{t_i}\| \leq V, V \in \mathbb{R}_+$. Such boundedness assumptions are standard (Theorem 1.1 in (Gao et al., 2024), Lemma 7 in (Mena et al., 2025)), in theoretical analyses of flow-based generative models and hold, for example, for rectified-flow velocity fields on bounded domains, where the velocity norm is controlled by the domain diameter. In practice, the assumption can also

be enforced by bounded network outputs or output clipping. With eq. (48) and eq. (50), the upper bound of scheduled deficit d_i can be derived as

$$d_i \leq \frac{\bar{h} + GV}{N} \tag{51}$$

With the non-zero gradient $\|\nabla h(\hat{\boldsymbol{\tau}}_{t_i})\| \geq m, m \in \mathbb{R}_+$, the minimum correction of $\boldsymbol{u}_{t_i}$ by eq. (47) can be written as

$$||\boldsymbol{u}_{t_i}\Delta t|| = \frac{d_i}{\|\nabla h(\hat{\boldsymbol{\tau}}_{t_i})\|} \leq \frac{\bar{h} + GV}{mN} \tag{52}$$

since $\Delta t = \frac{1}{N}$, we can finally obtain $\bar{U}$ as

$$||\boldsymbol{u}_{t_i}|| \leq \bar{U}, \quad \bar{U} = \frac{\bar{h} + GV}{m}, \tag{53}$$

which can be estimated by the property of constraints $h(\cdot)$ and MeanFlow model $\boldsymbol{\mu}^{\boldsymbol{\theta}}_{t_i}$.

C Additional Evaluation

C.1 Performance Across Navigation, Locomotion, and Manipulation

We further evaluate ASAP against the following baselines across navigation, locomotion, and manipulation tasks:

- **Flow Matching (FM)** (Lipman et al., 2022) generates trajectories directly with the pretrained flow model without safety enforcement.
- **Truncation** Brockman et al. (2016) modifies diffusion-generated trajectories to satisfy safety constraints.
- **Rejection Sampling** samples a batch of candidate trajectories by a pretrained MeanFlow model and selects the safest one.

As shown in Table 7, ASAP is the only method that satisfies safety across all benchmarks while maintaining competitive task performance. Flow matching sometimes achieves the highest score of the task, which is benefited by its multi-step sampling, but provides no safety guarantee. Truncation produces safe trajectories on some tasks but still lacks guarantees and requires multi-step sampling, inducing a higher computational cost. Rejection sampling can be combined with MeanFlow for efficient generation, yet cannot reliably satisfy unseen test-time constraints because safety depends on sufficiently safe candidates from the learned distribution. Overall, these results demonstrate that ASAP uniquely combines efficient planning with consistent safety across diverse robotic tasks.

Table 7. Comparison with other baselines across navigation, locomotion, and manipulation. Safety violation (s-vio), planning time (s) and task score of Flow matching (FM), Truncation, Reject sampling and ASAP on Maze2d, Hopper and KUKA block stacking (the 30% constraint level).

Experiment	Metric	FM	Truncation	Reject sampling	ASAP
Maze2d (Large)	s-vio (↓)	0.41 ± 0.34	0.07 ± 0.15	0.15 ± 0.23	$\mathbf{0.00 \pm 0.00}$
	time (↓)	2.86 ± 0.04	5.43 ± 2.94	$\mathbf{0.01 \pm 0.00}$	0.06 ± 0.01
	score (↑)	$\mathbf{1.61 \pm 0.01}$	1.50 ± 0.25	1.37 ± 0.51	1.46 ± 0.47
Hopper (ME)	s-vio (↓)	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
	time (↓)	0.12 ± 0.00	0.15 ± 0.00	0.04 ± 0.00	$\mathbf{0.05 \pm 0.00}$
	score (↑)	$\mathbf{0.44 \pm 0.05}$	0.42 ± 0.08	0.43 ± 0.12	0.41 ± 0.14
KUKA Block Stacking (30% constraint)	s-vio (↓)	0.92 ± 0.17	$\mathbf{0.00 \pm 0.00}$	0.25 ± 0.06	$\mathbf{0.00 \pm 0.00}$
	time (↓)	9.70 ± 0.15	21.83 ± 0.83	0.06 ± 0.00	$\mathbf{0.10 \pm 0.00}$
	score (↑)	1.05 ± 0.86	$\mathbf{1.25 \pm 1.09}$	1.10 ± 0.83	1.10 ± 0.89

Table 8. Adroit Relocate evaluation for all baselines. Safety violation, planning time (s), and task score on dexterous manipulation with a 39-dimensional state space and a 30-dimensional action space.

Task	Metric	MF	SFM	SAD-F	Diffuser	CG	S-Diff.	CoBL	ASAP
Adroit (Relocate)	s-vio (↓)	0.36±0.07	0.12±0.00	**0.00±0.00**	0.01±0.02	0.12±0.00	**0.00±0.00**	0.02±0.03	**0.00±0.00**
	time (↓)	**0.02±0.00**	0.28±0.00	1.54±0.15	0.13±0.00	0.14±0.00	0.59±0.01	0.16±0.01	0.10±0.00
	score (↑)	0.88±0.37	0.18±0.26	1.05±0.23	0.91±0.41	0.78±0.39	**1.07±0.08**	0.01±0.02	0.90±0.33

Table 9. Real-world handover for all baselines. Safety violation (s-vio), planning time (s), and task score over 20 trials.

Task	Metric	MF	SFM	SAD-F	CG	RS	Diff.	S-Diff.	ASAP
Handing ice cream	s-vio (↓)	0.35±0.11	0.01±0.00	**0.00±0.00**	0.27±0.12	0.28±0.08	0.27±0.13	**0.00±0.00**	**0.00±0.00**
	time (↓)	**0.01±0.01**	2.31±0.26	2.28±0.27	2.18±0.23	**0.01±0.01**	0.86±0.08	13.23±1.35	0.04±0.01
	score (↑)	**1.00±0.01**	0.74±0.43	**1.00±0.01**	0.99±0.01	0.89±0.30	0.95±0.22	0.94±0.22	**1.00±0.01**

C.2 Performance on High-Dimensional Tasks

We evaluate ASAP and all baselines on the Adroit Relocate task, a high-dimensional dexterous manipulation benchmark in which a robotic hand picks and relocates a ball. The task has a 39-dimensional state space and a 30-dimensional action space. As shown in Table 8, ASAP is the fastest method among all methods that guarantees safety. SAD-Flower and SafeDiffuser also satisfy the safety constraints and achieve higher task scores, but incur substantially greater computational cost due to repeated multi-step prediction and optimization. Other baselines violate the safety constraints and are also slower than ASAP because they require multiple sampling steps. MeanFlow is capable of planning faster marginally but cannot guarantee safety. These results demonstrate that ASAP maintains safety with low computational cost even in high-dimensional robotic tasks.

C.3 Real-World Robotic Planning Results

We provide the complete evaluation of the real-world robotic planning experiment introduced in section 5. The task requires a Franka robot to pick an ice cream from a plate and deliver it through a shop window to a target hand position while avoiding a juicer placed as an unseen obstacle. Demonstrations are collected by a teleoperation system without this test-time constraint, and deployment assumes that the ice-cream and target hand positions are provided by perception. We evaluate 20 trials across two initial plate locations and three target hand positions. We assume a perception system, e.g., motion capture or a depth camera, provides accurate object states. In our deployment, we manually specify the ice cream and target hand locations.

We evaluate each method in terms of safety, planning time, and task performance. Safety is measured by the minimum collision margin along the generated trajectory, normalized to $[0, 1]$. Task performance is determined by the final placement error, with a score of 1 for delivery within 1.0 cm of the target and higher scores for greater accuracy. We compare ASAP with MeanFlow (MF), Classifier Guidance (CG), and SafeDiffuser (S-Diff.), representing strong baselines in planning efficiency, task performance, and safety, respectively, and additionally with Diffuser (Diff.), SafeFlowMatcher (SFM), and SAD-Flower (SAD-F). We introduce an additional baseline, Rejection-Sampling (RS) based on MeanFlow, which samples a batch of trajectory candidates with MeanFlow, and selects the one that is closest to the safe set. RS can achieve fast planning due to a pretrained MeanFlow model, but cannot guarantee the safety since it fully relies on the dataset to include safety constraints.

The complete results are reported in Table 9. Across the evaluated configurations, ASAP provides the best overall combination of safety, planning efficiency, and task performance. MF achieves fast planning but cannot reliably satisfy the unseen obstacle constraint, while CG maintains strong task performance yet similarly lacks safety guarantees. Both of the S-Diff and SAD-Flower satisfy the safety constraint but require substantially longer planning time due to their multi-step safety corrections. SafeFlowMatcher relies on its time-varying parameter settings and can generate unsafe trajectories sometimes. In contrast, ASAP consistently maintains collision-free trajectories while preserving high task performance and low planning latency. These results further demonstrate that the arbitrary-step safety formulation transfers effectively from simulation to real-world manipulation under unseen constraints.

Table 10. The computational cost of ASAP with varying horizon length on Hopper.

	$H = 20$	$H = 40$	$H = 60$	$H = 80$	$H = 100$
time (s)	0.034 ± 0.001	0.037 ± 0.000	0.040 ± 0.001	0.041 ± 0.000	0.042 ± 0.000

Table 11. The computational cost of ASAP with varying amounts of constraints N_C on Maze2d.

	$N_C = 1$	$N_C = 2$	$N_C = 4$
time (s)	0.056 ± 0.002	0.056 ± 0.007	0.063 ± 0.003

C.4 Computational Scalability of the QP.

ASAP formulates a control-augmented MeanFlow and computes the control through a quadratic program (QP). To analyze the factors affecting QP computation time, we vary the trajectory horizon in Hopper, which increases the number of optimization variables, and the number of safety constraints in Maze2d. All evaluations use the same hardware and software settings as the main experiments in Section 5. As shown in Table 10 and Table 11, increasing either the trajectory horizon or the number of constraints increases the QP solving time only modestly. These results demonstrate that the computational cost of ASAP scales favorably with both trajectory dimensionality and constraint number, supporting its applicability to more complex planning tasks.

C.5 Practical Evaluation of Conservatism

To evaluate the practical effect of the zero conservatism established in Theorem 4.2, we examine whether ASAP avoids unnecessary clearance from the safety boundary across $N \in 1, 2, 3, 4, 5$ in Maze2d. We quantify this using a geometric safety margin (sm.), defined as the mean distance to the safety boundary over the 10 trajectory points closest to it, where smaller values indicate less conservative clearance. As shown in Figure 4, the margin remains consistently small and nearly unchanged across N, indicating that few-step generation does not introduce excessive geometric conservatism. Meanwhile, the score varies only slightly, showing that task performance is also insensitive to the number of generation steps.

D Environment Details

We evaluate ASAP across diverse trajectory-planning domains, summarized in table 12: navigation (Maze2d-Large), locomotion (Hopper), robotic manipulation (KUKA Block-Stacking), and dexterous manipulation (Adroit Relocate). The environments are simulated in MuJoCo (Todorov et al., 2012) or PyBullet (Coumans & Bai, 2016–2021) and span different state and action dimensions. For Maze2d, Hopper, and KUKA, we follow the default planning horizon and sampling steps in (Xiao et al., 2025). For Adroit, we use a shorter horizon and the same number of sampling steps as Hopper, since both jointly plan states and actions.

We train the generative and control models using offline datasets summarized in table 13. The navigation, locomotion, and Adroit datasets are obtained from D4RL (Fu et al., 2020), with dataset variants such as medium and medium-expert. Adroit Relocate data are generated by a fine-tuned RL policy (Rajeswaran et al., 2017), while Hopper uses soft actor-critic (SAC) trajectories (Haarnoja et al., 2018) mixed with expert demonstrations. For KUKA Block-Stacking (Janner et al., 2022), trajectories are generated using PDDLStream (Garrett et al., 2020), which provides feasible manipulation plans for structured stacking tasks.

E Test-Time Safety Constraints

We evaluate all methods under safety constraints that are introduced only at test time and are not represented in the training datasets.

Maze2d-Large. The planner generates a trajectory from a randomly sampled initial state to a randomly sampled goal while navigating the maze. To evaluate unseen constraint handling, we introduce two circular obstacles at test time that restrict the feasible region without fully blocking

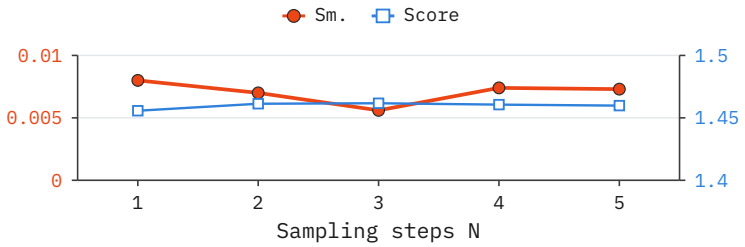

Figure 4. Safety margin (sm.) across sampling steps in Maze2d. left axis: geometry safety margin to the obstacle boundary; blue, right axis: task score. Both flat from $N = 1$ to 5.

Table 12. Benchmark settings.

Environment	Evaluation trials	Obs. Dim.	Action Dim.	Horizon	Default Sampling Steps	ASAP Sampling Steps
Maze2d-Large-v1	10	4	2	384	256	1
Hopper (Medium-Expert)	10	11	3	600	20	1
KUKA Block-Stacking	30	39	-	128	1000	4
Adroit Relocate (Expert)	20	39	30	32	20	3
Real-world handover	20	8	8	128	256	2

the path. Each obstacle is represented by the SDF

$$h(x, y) = \sqrt{(x - c_x)^2 + (y - c_y)^2} - r, \tag{54}$$

where $(x, y) \in \mathbb{R}^2$ is the planar state, (c_x, c_y) is the obstacle center, and $r > 0$ is its radius. Safety requires $h(x, y) \geq 0$.

Hopper. For locomotion, we impose an unseen torso-height constraint to emulate overhead obstacles. Let s denote the torso height. Safety requires

$$s \leq z, \tag{55}$$

with $z = 1.6$. This constraint conflicts with the forward-locomotion objective, making Hopper particularly challenging because satisfying safety can reduce task performance.

KUKA Block-Stacking. For KUKA manipulation, the generative model is conditioned on object locations and predicts joint-space trajectories. We impose tighter joint limits at test time to emulate workspace restrictions or partial system degradation:

$$\boldsymbol{q} \in \beta[\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}], \tag{56}$$

where $\beta \in \{0.9, 0.8, 0.7\}$ controls the constraint severity, which maps to the constraint level $= \{10\%, 20\%, 30\%\}$ in the Figure 3(b). Smaller β values progressively restrict the feasible joint range and allow us to evaluate robustness under increasingly strict unseen constraints.

Adroit Relocate. For dexterous manipulation, the policy is conditioned on the initial object position and relocation target and jointly plans state-action trajectories. To emulate an unsafe workspace region, we impose the unseen Cartesian constraint

$$x \leq 0.1, \tag{57}$$

which restricts the arm workspace during relocation.

Together, these constraints evaluate whether each method can satisfy previously unseen safety requirements while preserving task performance across navigation, locomotion, and manipulation.

F Training Details

We provide the implementation and training details for all evaluated methods, including codebases, optimization settings, and model architectures.

Table 13. Datasets. Size and source or generation method of the training data for each benchmark.

Environment	# of Trajectories	Source / Generation Method
Maze2d-Large-v1	4×10^6	D4RL (Fu et al., 2020)
Hopper (Medium-Expert)	2×10^6	Mixture of expert and partial SAC (Haarnoja et al., 2018)
KUKA Block-Stacking	10,000	PDDLStream planner (Garrett et al., 2020)
Adroit Relocate (Expert)	1×10^6	Finetuned RL policy (Rajeswaran et al., 2017)

Table 14. Training configurations. Batch size, learning rate and training steps of every method on each benchmark.

Maze2d-Large	Diffuser	CG	S-Diff.	FM	SAD-F	SFM	CoBL	ASAP
Batch Size	128	128	128	128	128	128	128	128
Learning Rate	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$4e^{-4}$
Steps	$5e^5$	$5e^5$	$5e^5$	$2.5e^5$	$2.5e^5$	$5e^5$	$5e^5$	$2.5e^5$
Hopper	Diffuser	CG	S-Diff.	FM	SAD-F	SFM	CoBL	ASAP
Batch Size	128	128	128	128	128	128	128	128
Learning Rate	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$4e^{-4}$
Steps	$2.5e^5$	$2.5e^5$	$2.5e^5$	$2.5e^5$	$2.5e^5$	$2.5e^5$	$2.5e^5$	$2.5e^5$
KUKA Block-Stacking	Diffuser	CG	S-Diff.	FM	SAD-F	SFM	CoBL	ASAP
Batch Size	64	64	64	64	64	64	64	64
Learning Rate	$2e^{-5}$	$2e^{-5}$	$2e^{-5}$	$2e^{-4}$	$2e^{-4}$	$2e^{-5}$	$2e^{-5}$	$1e^{-4}$
Steps	$7e^5$	$7e^5$	$7e^5$	$7e^5$	$7e^5$	$7e^5$	$7e^5$	$7e^5$
Adroit Relocate	Diffuser	CG	S-Diff.	FM	SAD-F	SFM	CoBL	ASAP
Batch Size	64	64	64	64	64	128	64	128
Learning Rate	$2e^{-5}$	$2e^{-5}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$
Steps	$1e^6$	$1e^6$	$1e^6$	$1e^6$	$1e^6$	$2.5e^6$	$1e^6$	$5e^5$

Implementations. We implement each baseline using its official or publicly available codebase whenever possible. Diffuser follows Janner et al. (2022); Classifier Guidance and SafeDiffuser followXiao et al. (2025); SAD-Flower follows Huang et al. (2026); SafeFlowMatcher follows Yang et al. (2026) and Flow Matching follows Feng et al. (2025). We implement CoBL with the algorithm described in Mizuta & Leung (2024) and the rejection sampling method based on the pretrained MeanFlow model. For our method, we consider the codebase from Flow matching Feng et al. (2025), SAD-Flower Huang et al. (2026), MeanFlow Geng et al. (2025), and the MeanFlow policy Sheng et al. (2026) and follow a similar framework as Feng et al. (2025), enabling a direct comparison under a shared generative backbone.

Training Configuration. The training hyperparameters used for each environment are summarized in Table 14. Within each environment, all methods use the same planning horizon. Our method uses the similar amount of training step as other baseline, so the comparison isolates the effect of our proposed safety mechanism rather than differences in model training.

Model Architecture. The diffusion-based baselines, including Diffuser, Classifier Guidance, CoBL, and SafeDiffuser, adopt the temporal U-Net architecture of Janner et al. (2022), with residual temporal convolutions, group normalization, and Mish activations. The SafeFlowMatcher also applies a temporal U-Net architecture following its source code. For flow-based baselines, including Flow Matching, SAD-Flower, and our method use the Transformer backbone of Feng et al. (2025), with a hidden dimension of 256, eight Transformer layers, eight attention heads, and an MLP intermediate dimension of 1024.

G Computational Resources

All simulation training and evaluation runs were conducted on a single platform with an AMD EPYC processor and an NVIDIA RTX 4090 GPU with 24 GB of memory. Training for the real-world robot-handing task used the same platform, while evaluation was performed on a separate local machine. A complete summary of the experimental hardware is provided in table 15.

Table 15. Compute. Hardware used for all training and evaluation for all experiments.

Usage	CPU	GPU	GPU RAM
Simulation task training	AMD EPYC 7313P	NVIDIA RTX 4090	24 GB
Simulation task evaluation	AMD EPYC 7313P	NVIDIA RTX 4090	24 GB
Real-world task training	AMD EPYC 7313P	NVIDIA RTX 4090	24 GB
Real-world task evaluation	AMD Ryzen 7950X3D	NVIDIA RTX 4080	16 GB

Across Maze2d, locomotion, KUKA, Adroit, and the real-world robot-handing task, the safety-guaranteed baselines, including SafeFlowMatcher (Yang et al., 2026), SAD-Flower (Huang et al., 2026), and SafeDiffuser (Xiao et al., 2025), require QP optimization as ASAP does. For a fair planning-time comparison, all methods use the same QP solver, *qpth* (Amos & Kolter, 2017). Because *qpth* supports only QP formulations, the result of post-optimization ablation in Table 4, which involves a non-QP problem, instead evaluates both ASAP and the post-optimization baseline using the package in *SciPy*.