HardFlow: Hard-Constrained Sampling for Flow-Matching Models via Trajectory Optimization

Zeyang Li, Kaveh Alim, Navid Azizan

Massachusetts Institute of Technology

***Abstract*—Diffusion and flow-matching have emerged as powerful methodologies for generative modeling, with remarkable success in capturing complex data distributions and enabling flexible guidance at inference time. Many downstream applications, however, demand enforcing hard constraints on generated samples—for example, robot trajectories must avoid obstacles—a requirement that goes beyond simple guidance. Prevailing projection-based approaches constrain the entire sampling path to the constraint manifold, which is overly restrictive and degrades sample quality. In this paper, we introduce a novel framework that reformulates hard-constrained sampling as a trajectory optimization problem. Our key insight is to leverage numerical optimal control to steer the sampling trajectory so that constraints are satisfied precisely at the terminal time. By exploiting the underlying structure of flow-matching models and adopting techniques from model predictive control, we transform this otherwise complex constrained optimization problem into a tractable surrogate that can be solved efficiently and effectively. Furthermore, this trajectory optimization perspective offers significant flexibility beyond mere constraint satisfaction, allowing for the inclusion of integral costs to minimize distribution shift and terminal objectives to further enhance sample quality, all within a unified framework. We provide a control-theoretic analysis of our method, establishing bounds on the approximation error between our tractable surrogate and the ideal formulation. Extensive experiments across diverse domains, including robotics (planning), partial differential equations (boundary control), and vision (text-guided image editing), demonstrate that our algorithm, which we name *HardFlow*, substantially outperforms existing methods in both constraint satisfaction and sample quality.**

***Index Terms*—Flow matching, diffusion, training-free guidance, constrained sampling, constrained optimization, optimal control, trajectory optimization.**

I. Introduction

Diffusion [1], [2] and Flow-matching [3] have revolutionized generative modeling, achieving tremendous performance across diverse domains, including image synthesis [4], [5], video generation [6], [7], molecule design [8], and robotics [9], [10]. At their core, these methods learn a time-varying "dynamics" (e.g., score function, velocity field) that transports samples from a simple prior distribution (e.g., Gaussian noise) to a complex target data distribution. The training process is notably efficient, leveraging conditional probability paths to formulate a conditional loss that enables simulation-free learning. During inference, samples are drawn from the simple prior and then evolved along trajectories defined by the learned dynamics, a process governed by an ordinary differential equation (ODE) or a stochastic differential equation (SDE). Terminal states of the trajectories constitute samples from the target distribution. A key feature of these models is their flexibility, which allows for guidance during inference to steer samples toward regions of interest, typically by injecting the gradient of a differentiable objective into the learned dynamics [11], [12].

Corresponding author: Zeyang Li (zeyang@mit.edu).

While gradient-based guidance is effective for encouraging desirable *soft* attributes, many practical applications present a more stringent challenge: enforcing *hard* constraints, i.e., inviolable conditions that a sample must satisfy to be considered valid. For example, in robotic planning, a proposed trajectory must be collision-free or it may lead to catastrophic failures; in facial-expression image editing, the subject's identity should remain invariant as the expression changes. Moreover, recent work has also highlighted the utility of explicit hard constraints as an effective mechanism to prevent reward overoptimization in text-to-image diffusion models [13].

To date, the predominant approach to hard-constrained sampling for diffusion and flow-matching models has been projection-based methods and their variants [14]–[22]. The most straightforward application of projection is a single, post-hoc correction where the final generated sample is projected onto the feasible set [14]. This simple approach, however, can often induce a significant shift away from the target data distribution [15]. To mitigate this, common refinements involve applying a projection within the sampling process. One prominent strategy is to project iteratively at each step, enforcing feasibility throughout the entire generation path [15], [16], [20], [21]. An alternative method is to ignore or relax the projections in early sampling steps, motivated by the insight that early-stage samples resemble noise and that tightly constraining these noisy precursors can be detrimental to final sample quality [17]–[19], [22].

Despite some empirical success, projections during sampling suffer from several fundamental limitations. First, these methods are intrinsically conservative as they enforce pathwise feasibility, a condition that is unnecessarily restrictive since only the terminal state (i.e., the sample of interest) must satisfy the constraint. The intermediate iterates are more of algorithmic artifacts. By imposing constraints on these transient states, projection during sampling prematurely prunes the search space, which can prevent the generative process from discovering higher-quality solutions that would otherwise be accessible. Although per-step projection can be interpreted as projected gradient ascent on the data log-likelihood [15], this view relies on Langevin-style sampling with specific noise and step-size schedules and does not generally extend to

arbitrary SDE discretizations or the ODE-based samplers used in flow matching. Second, these projection-based methods solely address constraint satisfaction and are not designed to optimize a reward function. In practical applications, however, it is often desirable not only to enforce hard constraints but also to optimize rewards to improve sample quality. An ideal method should jointly address both aspects and produce feasible samples of high quality.

To address the aforementioned challenges, we propose a novel framework that reformulates hard-constrained sampling as a trajectory optimization problem. While in this work we focus our formulation and analysis on ODE sampling for flow-matching models, the methodology is applicable to diffusion models under ODE sampling as well, such as denoising diffusion implicit models (DDIM) [23]. We consider perturbations to the learned velocity field as control inputs to steer the sampling trajectory. Constraints are then imposed solely on the terminal state, providing a minimally invasive correction that ensures the final sample lies within the feasible set without unnecessarily constraining the path. Furthermore, this framework naturally accommodates the inclusion of auxiliary objectives. We consider two primary types: (i) integral costs that penalize control effort to minimize the distributional shift from the uncontrolled sampler, and (ii) terminal costs designed to promote desirable attributes in the final sample. The trajectory optimization perspective thus offers a principled and unified framework that jointly addresses constraint satisfaction, distribution consistency, and sample quality.

We note that prior work has connected diffusion and flow-matching models with optimal control for training-free guidance [24], [25]. However, these approaches focus exclusively on guiding the samples toward high-reward regions and do not address hard constraints. Methodologically, they are grounded in Pontryagin's Maximum Principle (PMP) [26], the cornerstone of *indirect* methods in optimal control [27]. This class of techniques is notoriously difficult to apply to problems with state constraints, thereby hindering the extension of [24], [25] to hard-constrained sampling. In contrast, our work adopts *direct* methods [27] such as collocation [28] to discretize the dynamics and formulate a constrained optimization problem, enabling the explicit enforcement of hard constraints.

Nonetheless, the optimization problem is particularly challenging to solve. First, the number of decision variables is the product of the data dimension and integration steps, which is computationally demanding. Second, terminal constraints have to be propagated backward through equations involving the neural network dynamics (i.e., the learned velocity field). This creates an exceptionally complex feasible set, causing off-the-shelf solvers to struggle to find a feasible solution, let alone an optimal one. To address these issues, we leverage the specific structure of flow-matching models and draw inspiration from model predictive control (MPC) [29] to derive a tractable surrogate, which leads to an efficient and scalable algorithm. The contributions of this paper are summarized as follows.

- We introduce a novel framework that reformulates hard-constrained sampling for flow-matching models as a trajectory optimization problem. This perspective departs from predominant projection-based methods by enforcing constraints solely at the terminal time, thereby avoiding unnecessary restrictions on the sampling path. The framework also accommodates auxiliary objectives to minimize distribution shift and enhance sample quality.
- We develop a scalable algorithm, named HardFlow, to solve the formulated problem through principled transformations. First, with MPC principles, we decompose the long-horizon problem into a sequence of single-step subproblems. This is enabled by the flow-matching structure, which allows us to formulate effective proxies for terminal costs and constraints with the posterior mean. Second, we introduce a reverse reparameterization that changes the decision variable from the current state to the predicted terminal state, thereby avoiding the implicit, highly complex feasible set induced by propagating terminal constraints backward through the neural dynamics. This reparameterization is mathematically equivalent to a single-step fixed-point iteration that approximates an inverse mapping.
- We provide a control-theoretic analysis of our method, quantifying the approximation error introduced by our tractable surrogate relative to the original problem. These results clarify the algorithm's objective and how it balances computational efficiency with formulation optimality.
- We conduct extensive experiments across diverse domains, including robotics (planning), partial differential equations (boundary control), and vision (text-guided image editing). Our method consistently outperforms existing approaches in both constraint satisfaction and sample quality.

II. Related Work

This section situates our work within the relevant literature. We start by covering prior work in detail on constrained sampling for diffusion and flow-matching models—the core challenge addressed in this paper. As the advantages and limitations were comprehensively discussed in the Introduction, we do not repeat them here. Next, we provide a brief overview of optimal control taxonomy, as our approach is grounded in this field. We then discuss existing approaches that exploit optimal control for generative modeling. Finally, we broaden the scope to training-free guidance, since constrained sampling can be viewed as a specific instance with stricter requirements.

Constrained sampling methods can be broadly categorized by how strictly they enforce constraints in the formulation. A common approach for *soft-constrained sampling* is to formulate the constraint as a penalty term within a reward function, using its gradient to guide the generation process. Mizuta et al. [30] design rewards based on control barrier functions (CBFs) to promote collision avoidance in diffusion-based robot control. Carvalho et al. [31] construct cost functions that encode safe behaviors for motion planning and bias the sampler accordingly. For applications requiring strict feasibility, *hard-constrained sampling* is necessary. The predominant solution is projection-based methods, where samples are projected onto the feasible set, differing primarily in when

the projection is applied. A straightforward approach is *post-hoc projection*. Power et al. [14] propose a post-processing step after diffusion sampling to ensure feasibility. Christopher et al. [15] demonstrate that such post-processing can cause the generated samples to diverge significantly from the original data distribution. To mitigate this, they propose *per-step projection* during Langevin-style diffusion sampling, which is theoretically interpreted as projected gradient ascent on the data log-likelihood. Römer et al. [16] use per-step projection for diffusion-based planning, along with model predictive control (MPC) to form closed-loop control. Zampini et al. [21] integrate per-step projection sampling with Stable Diffusion for text-to-image generation under stringent constraints. Luan et al. [20] use per-step projection for coupled generation tasks like creating image pairs. A less restrictive approach is *late-stage projection*, where corrections are applied only in the later steps of the sampling process. Yuan et al. [17] apply physics-based projections in late diffusion steps to improve motion plausibility. Bouvier et al. [22] use late-stage projections to ensure synthetic robot trajectories are dynamically feasible. Another less restrictive method is *per-step projection with early-stage relaxation*, allowing small violations in early steps while enforcing exact feasibility later. Zhang et al. [19] enforce constraints in the sampling process via a Lagrangian formulation. As multipliers start small, the constraints are effectively relaxed early on and become progressively tighter as the multipliers increase. Xiao et al. [18] incorporate a CBF into diffusion sampling for safe planning. The CBF condition permits violations before the sample enters the feasible set, thereby relaxing constraints in the early stages.

Optimal control taxonomy. Optimal control provides a framework for computing control policies that optimize a performance criterion while respecting system dynamics and constraints [26]. We distinguish *global* and *local* formulations. A global formulation seeks an optimal closed-loop policy for every state in the admissible state space and is classically obtained by solving the Hamilton-Jacobi-Bellman (HJB) equation. A local formulation fixes an initial state and computes an optimal state-control trajectory, yielding an open-loop solution. For local problems, there are two primary families of methods: *indirect* and *direct* [27]. Indirect methods follow an "optimize-then-discretize" approach. They first derive the first-order necessary conditions for the continuous-time system, typically using Pontryagin's Maximum Principle (PMP). These conditions are then discretized and solved, often as a two-point boundary-value problem or with a gradient-based algorithm. However, state constraints make PMP notoriously hard to apply, as they create jumps in the costate at entry and exit times, introduce constrained arcs, and can cause singularity issues [32]. Direct methods, in contrast, use a "discretize-then-optimize" strategy. They transcribe the continuous-time problem into a finite-dimensional nonlinear optimization problem using techniques like shooting or collocation. The resulting optimization problem is then solved numerically. This formulation naturally accommodates constraints, making direct methods well-suited for such problems. An orthogonal classification is *deterministic* versus *stochastic* optimal control, which depends on whether the system dynamics contain stochastic elements [33]. In robotics, *trajectory optimization* is the preferred term for local optimal control and, because constraints are pervasive, it often implicitly signals a direct-method formulation, hence our title.

Optimal control in generative modeling. A growing line of work develops an optimal control viewpoint on sampling from unnormalized distributions [34]–[38]. For fine-tuning diffusion models, Domingo-Enrich et al. [39] propose an adjoint matching framework from indirect stochastic optimal control. They start from PMP and derive a refined loss that exhibits better convergence and stability; Zhao et al. [40] take a global optimal control perspective by approximately solving the HJB equation. Regarding training-free guidance, Rout et al. [24] propose a stylization method for text-to-image diffusion models grounded in PMP, and Wang et al. [25] derive PMP-based gradient for guiding flow-matching models. In this work, we address the problem of hard-constrained sampling through the lens of direct optimal control, a perspective that has been largely unexplored in generative modeling.

Training-free guidance. There has been extensive research on this topic since seminal works such as [11], [12]. A recurring theme is to exploit posterior information during sampling [41], [42]. We highlight some more recent gradient-based approaches from distinct viewpoints: optimization [43], variational inference [44], conditional and marginal paths [45], and optimal control [24], [25]. Additionally, there is a growing interest in handling non-differentiable objectives [46]–[48]. Our work contributes to this literature by enabling not only reward optimization but also enforcement of hard constraints at inference time.

III. Background

Consider a time-indexed family of random variables $X_t \in \mathbb{R}^d$ with density p_t for $t \in [0,1]$. Let $v : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ denote the marginal velocity field. The associated flow map $\Phi : [0,1] \times [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is defined as the solution of an ordinary differential equation (ODE),

$$\begin{cases} \frac{d}{d\tau}\Phi_{s\to\tau}(x) = v_\tau\left(\Phi_{s\to\tau}(x)\right) \\ \Phi_{s\to s}(x) = x, \end{cases}$$

where $\tau \in [s,t]$. The pushforward of initial density p_0 by the flow yields the marginal probability path $p_t = (\Phi_{0\to t})_\sharp p_0$, which satisfies the continuity equation

$$\partial_t p_t + \nabla \cdot (p_t v_t) = 0, \quad p_0.$$

We can sample from the terminal density p_1 by drawing $x_0 \sim p_0$ and integrating the ODE on $[0,1]$:

$$x_1 = \Phi_{0\to 1}(x_0) \sim p_1.$$

Flow matching [3] is an efficient, simulation-free method for training a neural velocity field $v_t^\theta(x)$ to approximate $v_t(x)$, given samples from the target distribution p_1. The key idea is to design a conditional probability path $\{p_{t|Z}\}_{t\in[0,1]}$ that bridges p_0 and p_1, with Z as the conditioning variable, typically $Z = (X_0, X_1)$. The law of Z, denoted $\pi_{0,1}$, is called the coupling. For each realization of Z, denote the conditional

velocity field as $v_{t|Z}(\cdot \mid Z)$. The marginal and conditional velocity fields are related by:

$$v_t(x) = \mathbb{E}_{Z\sim\pi_{0,1},X_t|Z\sim p_{t|Z}}\left[v_{t|Z}(X_t \mid Z)\middle|X_t = x\right].$$

The conceptual flow-matching loss compares $v_t^\theta(x)$ to marginal velocity field $v_t(x)$:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],X_t\sim p_t}\left[\left\|v_t^\theta(X_t) - v_t(X_t)\right\|_2^2\right],$$

which is intractable since $v_t(x)$ is unknown. The conditional flow-matching loss instead compares $v_t^\theta(x)$ to the conditional velocity field $v_{t|Z}(x \mid Z)$:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}[0,1],\ Z\sim\pi_{0,1},\\ X_t|Z\sim p_{t|Z}}}\left[\left\|v_t^\theta(X_t) - v_{t|Z}(X_t \mid Z)\right\|_2^2\right].$$

It is shown that $\nabla_\theta\mathcal{L}_{\mathrm{CFM}}(\theta) = \nabla_\theta\mathcal{L}_{\mathrm{FM}}(\theta)$, thus we can train $v_t^\theta(x)$ with $\mathcal{L}_{\mathrm{CFM}}(\theta)$.

To construct the conditional velocity field, a widely adopted choice is to use an affine conditional path

$$X_t \mid Z = \alpha_t X_1 + \beta_t X_0,$$

with scheduler (α_t, β_t) satisfying $\alpha_0 = 0$, $\alpha_1 = 1$, $\beta_0 = 1$, $\beta_1 = 0$. In this case, $p_{t|Z} = \delta_{\alpha_t X_1 + \beta_t X_0}$, and the corresponding conditional velocity is

$$v_{t|Z}(X_t \mid Z) = \dot\alpha_t X_1 + \dot\beta_t X_0,$$

for $X_t = \alpha_t X_1 + \beta_t X_0$.

IV. Problem Formulation

In this paper, we introduce a training-free method for hard-constrained sampling from a pretrained flow-matching model, $v_t^\theta(x)$. Our approach is designed to operate with fixed model parameters θ, which enforce constraints by steering the sampling process. We do not consider fine-tuning or inference-time parameter update, which are a separate line of work and typically require substantially more computation. We explain the problem formulation in this section.

Suppose the initial distribution is given by p_0. The terminal distribution under the flow map is $\bar\mu = (\Phi_{0\to1}^\theta)_\sharp p_0$, obtained by sampling $x_0 \sim p_0$ and integrating $\dot x_t = v_t^\theta(x_t)$ to $t = 1$. We call $\bar\mu$ the *nominal distribution*, produced by the uncontrolled sampler (no interference during sampling).

In practice, a nominal distribution may be insufficient, as its samples often fail to meet specific task requirements. Many settings impose *hard constraints*, denoted as $h(x) \le 0$, which are non-negotiable conditions that all samples must satisfy. For example, in real-world robotics, sampled trajectories must be collision-free, and sampled actions must respect the robot's physical limits. In addition to these strict requirements, we often aim to minimize *costs*, $C(x)$, such as the energy consumption of a robot trajectory. These costs are secondary. Although lower values are desirable, higher costs are acceptable when necessary to satisfy the primary hard constraints. Moreover, we prefer that the adjusted distribution stay close to the nominal one $\bar\mu$.

Remark 1. *We use the shorthand $h(x) \le 0$ to represent multiple constraints. In particular, if $h(x) = (h_1(x), \dots, h_m(x))$, the inequality is understood componentwise: $h_i(x) \le 0$ for all $i = 1, \dots, m$. Equality constraints $g(x) = 0$ are handled by two inequalities $g(x) \le 0$ and $-g(x) \le 0$.*

Concretely, given $\bar x_0 \sim p_0$, we seek a per-sample refinement procedure that steers each trajectory during ODE integration to a terminal state $\tilde x_1$ such that (i) the hard constraints $h(\tilde x_1) \le 0$ are satisfied, (ii) the cost $C(\tilde x_1)$ is typically low, and (iii) collectively, over draws of $\bar x_0$, the resulting terminal distribution $\tilde x_1 \sim \tilde\mu$ remains close to the nominal distribution $\bar\mu$.

This ODE-steering perspective naturally fits within local optimal control. We introduce a control input u_t to perturb the learned velocity field $v_t^\theta(x)$ during sampling. The continuous-time optimal control problem is presented as follows.

Problem 1 (Continuous-time formulation)**.**
Given initial state $\bar x_0 \sim p_0$, solve the following problem to obtain x_1 as the sample.

$$\begin{aligned}\min_{\{x_t,u_t\}_{t\in[0,1]}}\quad & C(x_1) + \lambda_{\mathrm{oc}}\textstyle\int_0^1 \frac{1}{2}\|u_t\|_2^2 dt\\ \text{s.t.}\quad & x_0 = \bar x_0,\\ & \dot x_t = v_t^\theta(x_t) + u_t,\\ & h(x_1) \le 0.\end{aligned} \tag{1}$$

The constraints in (1) have three components. The initial condition $x_0 = \bar x_0$ ensures that the controlled trajectory starts from the same point as the uncontrolled one, which is drawn from the initial distribution p_0. The dynamics $\dot x_t = v_t^\theta(x_t) + u_t$ describe how the state evolves under the influence of both the neural velocity field and the control input. The terminal constraint $h(x_1) \le 0$ enforces that the final state, which is the true sample of interest, satisfies the hard constraint. The objective function J consists of two terms: the terminal cost $C(x_1)$, which encourages desirable attributes in the sample, and an integral cost $\frac{\lambda_{\mathrm{oc}}}{2}\int_0^1\|u_t\|_2^2 dt$, which penalizes large control efforts to keep the controlled trajectory close to the nominal one.

The following sections will delve into both algorithmic and theoretical aspects. Algorithmically, solving (1) online for each sample is extremely challenging: the neural dynamics v_t^θ is highly nonlinear, and the terminal constraint $h(x_1) \le 0$, coupled with the dynamics, makes the feasible sets for u_t and x_t (for $t < 1$) even more complex. We therefore seek principled transformations of Problem 1 into a tractable surrogate that can be solved efficiently at sampling time while retaining the essential structure of the original formulation. Theoretically, we quantify the approximation errors incurred by our surrogate, so that the gap between the surrogate and the original formulation (1) is explicit and interpretable. These analyses clarify what the proposed algorithm optimizes and how it balances computational efficiency and formulation optimality.

V. Algorithm Design

In this section, we develop a scalable algorithm to solve the optimal control problem (1) efficiently and effectively. Fig. 1 illustrates the roadmap of problem transformations, which we explain in detail below.

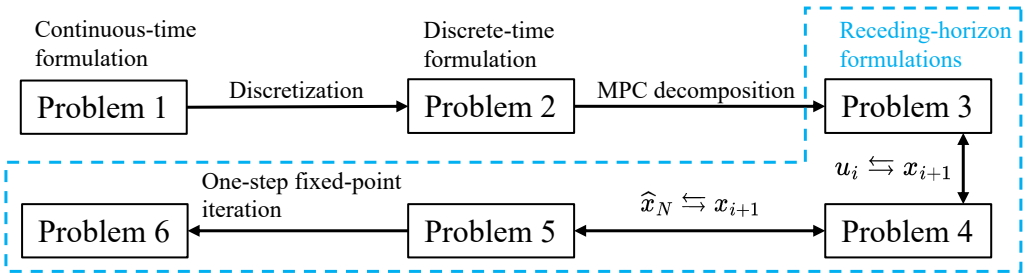

Fig. 1: Roadmap of problem transformations. Double arrows indicate equivalence; single arrows indicate approximation.

We begin by clarifying the scope. Problem 1 is a local formulation: for a fixed initial condition $x_0 = \bar{x}_0$, we seek an open-loop control u_t on $t \in [0,1]$. This stands in contrast to a global formulation that would compute an optimal feedback policy $u_t(x)$ for all states. The global problem is governed by the Hamilton-Jacobi-Bellman (HJB) equation and suffers from the curse of dimensionality. Data-driven approximations for HJB, such as reinforcement learning, require additional training and fall outside the scope of inference-time guidance.

Recent training-free guidance methods [24], [25] for diffusion and flow-matching models have followed the indirect ("optimize-then-discretize") route for local optimal control, making use of Pontryagin's Maximum Principle (PMP). However, as discussed in Related Work, indirect methods are notoriously difficult to apply to problems with state constraints, which is the core challenge we aim to address. This limitation motivates our adoption of a direct approach. We discretize the continuous-time dynamics and transcribe (1) into a finite-dimensional constrained optimization problem, about which we can reason comprehensively.

We discretize the continuous-time dynamics using the forward Euler method. This choice is made for notational simplicity without loss of generality, as the framework presented here extends to higher-order integrators. It is also consistent with standard sampling practices in flow-matching models [5], [39]. Define $0 = t_0 < t_1 < \cdots < t_N = 1$, and $\Delta t_j = t_{j+1} - t_j$ for $j = 0, 1, \cdots, N-1$. We arrive at the following problem.

Problem 2 (Discrete-time formulation)**.**
Given initial state $\bar{x}_0 \sim p_0$, solve the following problem to obtain x_N as the sample.

$$\begin{aligned} \min_{\{x_j\}_{j=0}^{N}, \{u_j\}_{j=0}^{N-1}} \quad & C(x_N) + \lambda_{\text{oc}} \textstyle\sum_{j=0}^{N-1} \frac{1}{2} \|u_j\|_2^2 \, \Delta t_j \\ \text{s.t.} \quad & x_0 = \bar{x}_0, \\ & x_{j+1} = x_j + v_{t_j}^{\theta}(x_j) \, \Delta t_j + u_j \Delta t_j, \\ & j = 0, 1, \cdots, N-1, \\ & h(x_N) \le 0. \end{aligned} \tag{2}$$

The decision variables in Problem 2 are the discrete states and controls, for a total of $2Nd + d$ scalars. Solving this problem directly is computationally demanding, especially for fine-grained discretizations (large N) or high-dimensional data such as images (large d). These challenges are compounded by the complex, multi-step coupling between states and controls via highly nonconvex neural network dynamics. To make the problem tractable, we employ a model predictive control (MPC) framework. This strategy decomposes the long-horizon optimization into a sequence of more manageable, short-horizon subproblems. The long-horizon coupling in Problem 2 arises from the terminal cost $C(x_N)$ and terminal constraint $h(x_N) \le 0$. Therefore, the key to applying MPC is to construct effective proxies for $C(x_N)$ and $h(x_N) \le 0$ at an intermediate state x_i. Fortunately, diffusion and flow-matching models possess a unique structure that facilitates this approximation. Specifically, they provide a posterior estimate of the terminal state x_N given an intermediate state x_i. For affine conditional paths, the following results hold.

Lemma 1 (Posterior mean [45])**.** *Let $Z = (X_0, X_1) \sim \pi_{0,1}$. Consider the affine conditional path $X_t \mid Z = \alpha_t X_1 + \beta_t X_0$ for $t \in [0,1]$ with differentiable scheduler (α_t, β_t). The conditional velocity field is therefore $v_{t|Z}(X_t \mid Z) = \dot{\alpha}_t X_1 + \dot{\beta}_t X_0$. Suppose $v_t(x)$ is the marginal velocity field, i.e., $v_t(x) = \mathbb{E}\left[v_{t|Z}(X_t \mid Z) \middle| X_t = x\right]$. Define $\Lambda_t := \alpha_t \dot{\beta}_t - \dot{\alpha}_t \beta_t$. Assume $\Lambda_t \neq 0$ for all $t \in [0,1]$. Then, for $x \in \mathbb{R}^d$, we have*

$$\mathcal{M}_t(x) := \mathbb{E}\left[X_1 \mid X_t = x\right] = \frac{\dot{\beta}_t x - \beta_t v_t(x)}{\Lambda_t},$$

$$\mathcal{N}_t(x) := \mathbb{E}\left[X_0 \mid X_t = x\right] = \frac{-\dot{\alpha}_t x + \alpha_t v_t(x)}{\Lambda_t}.$$

Additionally, the following identity holds:

$$x = \alpha_t \mathcal{M}_t(x) + \beta_t \mathcal{N}_t(x). \tag{3}$$

With Lemma 1, we can approximate the terminal state x_N given an intermediate state x_i by

$$\mathcal{M}_{t_i}^{\theta}(x_i) := \frac{\dot{\beta}_{t_i} x_i - \beta_{t_i} v_{t_i}^{\theta}(x_i)}{\Lambda_{t_i}},$$

for $i = 0, 1, \cdots, N$. In particular, $\mathcal{M}_{t_N}^{\theta}(x_N) = x_N$ under standard boundary conditions $\alpha_0 = \beta_1 = 0$ and $\alpha_1 = \beta_0 = 1$.

We now apply MPC to decompose Problem 2 into a sequence of one-step subproblems, yielding the following receding-horizon scheme.

Problem 3 (Receding-horizon formulation, original)**.**
Given initial state $\bar{x}_0 \sim p_0$, initialize $x_0 = \bar{x}_0$ and carry out the following recursion for $i = 0, 1, \cdots, N-1$ to obtain x_N as the sample.

$$\begin{cases} u_i^* = \underset{u_i}{\text{argmin}} \quad C(\widehat{x}_N) + \frac{\lambda_{\text{oc}}}{2} \|u_i\|_2^2 \, \Delta t_i \\ \qquad\qquad \text{s.t.} \quad x_{i+1} = x_i + v_{t_i}^{\theta}(x_i) \, \Delta t_i + u_i \Delta t_i, \\ \qquad\qquad\qquad\quad h(\widehat{x}_N) \le 0, \\ \qquad\qquad\qquad\quad \widehat{x}_N = \mathcal{M}_{t_{i+1}}^{\theta}(x_{i+1}). \\ x_{i+1} = x_i + v_{t_i}^{\theta}(x_i) \, \Delta t_i + u_i^* \Delta t_i \end{cases} \tag{4}$$

Problem 3 computes controls u_i step by step. Intuitively, u_i is chosen so that after it is applied to reach x_{i+1}, the resulting terminal state will be both low-cost and feasible, assuming no future controls are used. A direct approach would be to simulate the remaining trajectory from x_{i+1} to x_N at every step and then evaluate $C(x_N)$ and $h(x_N)$, which is computationally expensive. Instead, we use the posterior estimate $\widehat{x}_N = \mathcal{M}^{\theta}_{t_{i+1}}(x_{i+1})$ to approximate the terminal state. This captures the effect of u_i on the terminal outcome at an efficient $\mathcal{O}(1)$ cost per step. The control regularization $\frac{\lambda_{\text{oc}}}{2}\|u_i\|_2^2\Delta t_i$ discourages large deviations from the nominal dynamics.

From this perspective, Problem 3 is a principled approximation of Problem 2: it preserves the core structure while greatly improving tractability. The approximation errors arise from two sources: (i) the posterior estimate $\widehat{x}_N$ is not exact, and (ii) future controls $\{u_j\}_{j=i+1}^{N-1}$ are ignored when optimizing u_i. We will rigorously quantify these errors in the next section.

Observe that, since the control enters the dynamics additively, u_i can be viewed as a perturbation to the nominal next state $\bar{x}_{i+1} = x_i + v^{\theta}_{t_i}(x_i)\Delta t_i$. This motivates a change of variables from u_i to x_{i+1}, leading to an equivalent formulation as Problem 4. This transformation is beneficial as it clarifies the structure and enables further manipulations.

Problem 4 (Receding-horizon formulation, simplified)**.** *Given initial state $\bar{x}_0 \sim p_0$, initialize $x_0 = \bar{x}_0$ and carry out the following recursion for $i = 0, 1, \cdots, N-1$ to obtain x_N as the sample.*

$$\begin{cases} \bar{x}_{i+1} = x_i + v^{\theta}_{t_i}(x_i)\Delta t_i \\ x^*_{i+1} = \underset{x_{i+1}}{\operatorname{argmin}} \quad C(\widehat{x}_N) + \frac{\lambda_{\text{oc}}}{2\Delta t_i}\|x_{i+1} - \bar{x}_{i+1}\|_2^2 \\ \qquad\qquad \text{s.t.} \quad h(\widehat{x}_N) \le 0, \\ \qquad\qquad\quad \widehat{x}_N = \mathcal{M}^{\theta}_{t_{i+1}}(x_{i+1}). \\ x_{i+1} = x^*_{i+1} \end{cases} \tag{5}$$

Problem 4 involves two coupled states: the decision variable x_{i+1} and the predicted terminal state $\widehat{x}_N$. While the optimization is over x_{i+1}, the terminal cost and constraints are applied to $\widehat{x}_N$. The two states are linked by the mapping $\widehat{x}_N = \mathcal{M}^{\theta}_{t_{i+1}}(x_{i+1})$, which is highly nonlinear due to the embedded neural network $v^{\theta}_{t_{i+1}}(\cdot)$. Consequently, the feasible set for x_{i+1}, implicitly defined as $\left\{x_{i+1} \mid h\left(\mathcal{M}^{\theta}_{t_{i+1}}(x_{i+1})\right) \le 0\right\}$, is significantly complicated. This structure poses a significant challenge for numerical solvers, as it can be difficult to find even a single feasible point, regardless of the simplicity of $h(\cdot)$ (e.g., linear).

Since feasibility is of paramount importance, we aim to work directly with the original constraint $h(x_N) \le 0$, avoiding the complex distortion introduced by the neural network $v^{\theta}_{t_{i+1}}(\cdot)$. To this end, we propose a reversed formulation. We treat $\widehat{x}_N$ as the new decision variable and then express x_{i+1} in terms of $\widehat{x}_N$. Assuming the mapping $\mathcal{M}^{\theta}_{t_{i+1}}(\cdot)$ is invertible, we have the following formulation.

Problem 5 (Receding-horizon formulation, reversed)**.** *Given initial state $\bar{x}_0 \sim p_0$, initialize $x_0 = \bar{x}_0$ and carry out the following recursion for $i = 0, 1, \cdots, N-1$ to obtain x_N as the sample.*

$$\begin{cases} \bar{x}_{i+1} = x_i + v^{\theta}_{t_i}(x_i)\,\Delta t_i \\ \widehat{x}^*_N = \underset{\widehat{x}_N}{\operatorname{argmin}} \quad C(\widehat{x}_N) + \frac{\lambda_{\text{oc}}}{2\Delta t_i}\left\|(\mathcal{M}^{\theta}_{t_{i+1}})^{-1}(\widehat{x}_N) - \bar{x}_{i+1}\right\|_2^2 \\ \qquad\qquad \text{s.t.} \quad h(\widehat{x}_N) \le 0. \\ x_{i+1} = (\mathcal{M}^{\theta}_{t_{i+1}})^{-1}(\widehat{x}^*_N) \end{cases} \tag{6}$$

To make Problem 5 practical, we need to characterize the inverse mapping $(\mathcal{M}^{\theta}_{t_{i+1}})^{-1}(\cdot)$. From (3) in Lemma 1, for any x we have

$$x = \alpha_{t_{i+1}}\mathcal{M}^{\theta}_{t_{i+1}}(x) + \beta_{t_{i+1}}\mathcal{N}^{\theta}_{t_{i+1}}(x),$$

where

$$\mathcal{N}^{\theta}_{t_{i+1}}(x) = \frac{-\dot{\alpha}_{t_{i+1}}x + \alpha_{t_{i+1}}v^{\theta}_{t_{i+1}}(x)}{\Lambda_{t_{i+1}}}.$$

Then $y = \mathcal{M}^{\theta}_{t_{i+1}}(x)$ is equivalent to $x = \alpha_{t_{i+1}}y + \beta_{t_{i+1}}\mathcal{N}^{\theta}_{t_{i+1}}(x)$ for any y. Consequently, inverting $y = \mathcal{M}^{\theta}_{t_{i+1}}(x)$ amounts to finding a fixed point of the mapping

$$T^{y}_{t_{i+1}}(x) = \alpha_{t_{i+1}}y + \beta_{t_{i+1}}\mathcal{N}^{\theta}_{t_{i+1}}(x), \tag{7}$$

i.e., solving $x^* = T^{y}_{t_{i+1}}(x^*)$. Assuming $T^{y}_{t_{i+1}}(\cdot)$ is a contraction, the fixed point from any initial guess $x^{(0)}$ is the limit of the iterations

$$(\mathcal{M}^{\theta}_{t_{i+1}})^{-1}(y) = \lim_{k\to\infty}\left(T^{y}_{t_{i+1}}\right)^k(x^{(0)}). \tag{8}$$

In our setting, a natural initial guess is the nominal next state $\bar{x}_{i+1} = x_i + v^{\theta}_{t_i}(x_i)\,\Delta t_i$. Truncating the iterations after one step yields

$$(\mathcal{M}^{\theta}_{t_{i+1}})^{-1}(y) \approx T^{y}_{t_{i+1}}(\bar{x}_{i+1}) = \alpha_{t_{i+1}}y + \beta_{t_{i+1}}\mathcal{N}^{\theta}_{t_{i+1}}(\bar{x}_{i+1}).$$

This one-step cut-off is a deliberate design choice, presenting a linear approximation of $(\mathcal{M}^{\theta}_{t_{i+1}})^{-1}(y)$ that requires only a single forward pass of $v^{\theta}_{t_{i+1}}(\cdot)$ at $\bar{x}_{i+1}$. This eliminates the need to evaluate input gradients of $v^{\theta}_{t_{i+1}}(\cdot)$ within the associated optimization, greatly improving computational efficiency. The approximation error will be analyzed in the Theoretical Analysis section. We thus arrive at the final surrogate formulation as Problem 6.

Problem 6 (Receding-horizon formulation, surrogate)**.** *Given initial state $\bar{x}_0 \sim p_0$, initialize $x_0 = \bar{x}_0$ and carry out the following recursion for $i = 0, 1, \cdots, N-1$ to obtain x_N as the sample.*

$$\begin{cases} \bar{x}_{i+1} = x_i + v^{\theta}_{t_i}(x_i)\,\Delta t_i \\ \widehat{x}^*_N = \underset{\widehat{x}_N}{\operatorname{argmin}} \quad C(\widehat{x}_N) + \frac{\lambda_{\text{oc}}}{2\Delta t_i}\left\|\mathcal{F}^{\theta}_{t_{i+1}}(\widehat{x}_N) - \bar{x}_{i+1}\right\|_2^2 \\ \qquad\qquad \text{s.t.} \quad h(\widehat{x}_N) \le 0. \\ x_{i+1} = \mathcal{F}^{\theta}_{t_{i+1}}(\widehat{x}^*_N) \end{cases} \tag{9}$$

Specifically, $\mathcal{F}^{\theta}_{t_{i+1}} : \mathbb{R}^d \to \mathbb{R}^d$ is defined as

$$\mathcal{F}^{\theta}_{t_{i+1}}(\widehat{x}_N) = \alpha_{t_{i+1}}\widehat{x}_N + \beta_{t_{i+1}}\mathcal{N}^{\theta}_{t_{i+1}}(\bar{x}_{i+1}). \tag{10}$$

Since $\mathcal{F}^{\theta}_{t_{i+1}}(\cdot)$ is affine, we can further simplify the expressions. Evaluating the identity (3) at $\bar{x}_{i+1}$ gives

$\bar{x}_{i+1} = \alpha_{t_{i+1}} \mathcal{M}^{\theta}_{t_{i+1}}(\bar{x}_{i+1}) + \beta_{t_{i+1}} \mathcal{N}^{\theta}_{t_{i+1}}(\bar{x}_{i+1})$. Define $\bar{x}_N \coloneqq \mathcal{M}^{\theta}_{t_{i+1}}(\bar{x}_{i+1})$. Then, for any $\widehat{x}_N$,

$$\begin{aligned} \|x_{i+1} - \bar{x}_{i+1}\|_2^2 &= \left\| \mathcal{F}^{\theta}_{t_{i+1}}(\widehat{x}_N) - \bar{x}_{i+1} \right\|_2^2 \\ &= \left\| \alpha_{t_{i+1}} \widehat{x}_N - \alpha_{t_{i+1}} \mathcal{M}^{\theta}_{t_{i+1}}(\bar{x}_{i+1}) \right\|_2^2 \\ &= \alpha_{t_{i+1}}^2 \left\| \widehat{x}_N - \bar{x}_N \right\|_2^2 . \end{aligned}$$

Therefore, the optimization in (9) can be expressed entirely in terms of $\widehat{x}_N$, without involving x_{i+1}. We now summarize the complete workflow in Algorithm 1, which is named HardFlow.

Algorithm 1: HardFlow: Hard-constrained sampling for flow-matching models

Input: Initial distribution p_0, learned flow-matching model $v_t^{\theta}(\cdot)$, cost $C(\cdot)$, constraint $h(\cdot) \le 0$, regularization parameter $\lambda_{\mathrm{oc}} > 0$, discretization steps N, time grid $0 = t_0 < t_1 < \cdots < t_N = 1$, and differentiable affine scheduler (α_t, β_t).

1 Draw initial state $\bar{x}_0 \sim p_0$ and set $x_0 = \bar{x}_0$;

2 **for** $i = 0$ **to** $N - 1$ **do**

3 Compute $\Delta t_i = t_{i+1} - t_i$.

4 Compute $\bar{x}_{i+1} = x_i + v_{t_i}^{\theta}(x_i)\, \Delta t_i$.

5 Compute $\bar{x}_N = \frac{\dot{\beta}_{t_{i+1}} \bar{x}_{i+1} - \beta_{t_{i+1}} v^{\theta}_{t_{i+1}}(\bar{x}_{i+1})}{\alpha_{t_{i+1}} \dot{\beta}_{t_{i+1}} - \dot{\alpha}_{t_{i+1}} \beta_{t_{i+1}}}$.

6 Solve the optimization problem

$$\widehat{x}_N^* = \underset{\widehat{x}_N}{\operatorname{argmin}} \quad C(\widehat{x}_N) + \frac{\lambda_{\mathrm{oc}}}{2\Delta t_i} \alpha_{t_{i+1}}^2 \left\| \widehat{x}_N - \bar{x}_N \right\|_2^2 \quad \text{s.t.} \quad h(\widehat{x}_N) \le 0. \tag{11}$$

7 Compute $x_{i+1} = \alpha_{t_{i+1}} \widehat{x}_N^* + \beta_{t_{i+1}} \frac{-\dot{\alpha}_{t_{i+1}} \bar{x}_{i+1} + \alpha_{t_{i+1}} v^{\theta}_{t_{i+1}}(\bar{x}_{i+1})}{\alpha_{t_{i+1}} \dot{\beta}_{t_{i+1}} - \dot{\alpha}_{t_{i+1}} \beta_{t_{i+1}}}$.

8 **end**

Output: Sample x_N.

HardFlow is essentially equivalent to solving Problem 6, where the optimization (11) is written in an explicit $\widehat{x}_N$-only form. For completeness, we also expand intermediate expressions and present the final form.

It is worth emphasizing that, although we perform a series of transformations on Problem 2 to obtain Problem 6, there is no relaxation of terminal feasibility. More precisely, we have the following result.

Proposition 1. *Suppose the feasible set $\mathcal{S} = \{x \mid h(x) \le 0\}$ is nonempty. The output sample x_N of Algorithm 1 satisfies the hard constraints $h(x_N) \le 0$.*

Proof. At the final iteration $i = N - 1$, let $\widehat{x}_N^*$ denote a solution to (11). Then $h(\widehat{x}_N^*) \le 0$. Also note that $t_N = 1$. The boundary conditions of the affine scheduler (α_t, β_t) at $t = 1$ are $\alpha_1 = 1$ and $\beta_1 = 0$.

Therefore, for $i = N - 1$, we can simplify the update rule in Line 7 of Algorithm 1 as

$$x_N = \alpha_1 \widehat{x}_N^* + \beta_1 \frac{-\dot{\alpha}_1 \bar{x}_N + \alpha_1 v_1^{\theta}(\bar{x}_N)}{\alpha_1 \dot{\beta}_1 - \dot{\alpha}_1 \beta_1} = \widehat{x}_N^*. \tag{12}$$

Consequently, $h(x_N) = h(\widehat{x}_N^*) \le 0$, which shows that the output sample x_N satisfies the hard constraints. □

Remark 2. *In the unconstrained setting, where only a cost $C(\cdot)$ (equivalently, a reward $R(\cdot) = -C(\cdot)$) is present, the task reduces to guided sampling with a reward model, for which multiple formulations and algorithms exist in isolation but have not been unified. Problem 1 without constraints matches the optimal control formulation in [25], [39]. Solving Problem 4 without constraints by gradient descent is akin to the gradient-based methods in [43]–[45], which differentiate the predicted state with respect to the current state. Problem 6 without constraints is in the same spirit as [49] and the proxy formulation in [24], both of which employ DDIM-like updates that jump to the terminal state, optimize it, and map back to the intermediate state, whereas our flow-matching approach is more general since it does not require a Gaussian source.*

To the best of our knowledge, the hierarchy from Problem 1 to Problem 6 proposed in this paper is the first to systematically connect these formulations through the lens of optimal control. Starting from the continuous-time Problem 1, discretization, MPC decomposition, a control-to-state change of variables, a reverse reparameterization, and a one-step fixed-point iteration link all the formulations in a principled manner. This unified view, together with the theoretical analysis in the next section, may be of independent interest for reward-only scenarios.

Remark 3. *The constrained optimization problem* (11) *can be solved using any suitable method, depending on the specific downstream application. In general, it may be solved numerically with constrained optimization algorithms, such as sequential quadratic programming or interior-point methods. Moreover, if the cost $C(\cdot)$ and constraints $h(\cdot)$ possess particular structures (e.g., a quadratic cost and linear constraints), specialized solvers or even closed-form solutions may be available.*

In summary, this section utilizes trajectory optimization to enforce hard constraints and minimize costs on samples at inference time. Starting from the continuous-time formulation (Problem 1), we discretize it (Problem 2) and distill it (Problem 3 to 6) into a tractable algorithm via two major transformations: (i) an MPC decomposition that reduces the full-horizon problem to sequential one-step subproblems, and (ii) a reversed formulation with a single-step fixed-point iteration that sidesteps complex feasible sets and improves computational efficiency. We will provide comprehensive theoretical justifications in the Theoretical Analysis section and demonstrate the practical effectiveness of our algorithm through extensive evaluations in the Experiments section.

VI. Theoretical Analysis

In this section, we present a detailed analysis of the proposed algorithm. Leveraging control-theoretic tools, we quantify the approximation errors introduced by the two major transformations used to derive Problem 6 from Problem 2.

First, we analyze the suboptimality introduced by the MPC decomposition, i.e., the gap between Problem 2 and Problem 3. For notation convenience, define the stage utility $l_i(u) \coloneqq \frac{\lambda_{\mathrm{oc}}}{2} \|u\|_2^2 \Delta t_i$, the (Problem 2) objective $\mathcal{J} \coloneqq$

$C(x_N) + \sum_{j=0}^{N-1} l_j(u_j)$, the one-step transition $\Psi_i^\theta(x,u) \coloneqq x + v_{t_i}^\theta(x)\Delta t_i + u\Delta t_i$ with $\Psi_i^\theta(x) \coloneqq \Psi_i^\theta(x,0)$, and the terminal feasible set $\mathcal{S} = \{x \mid h(x) \le 0\}$.

We assume that, given an initial state x_0, both Problem 2 and Problem 3 admit feasible solutions. We further define the per-step feasible sets for Problem 3 as $\widehat{\mathcal{S}}_i = \{x \mid h(\mathcal{M}_{t_i}^\theta(x)) \le 0\}$, for $i = 1, 2, \cdots, N$. Since $\mathcal{M}_{t_N}^\theta(x) = x$, we have $\widehat{\mathcal{S}}_N = \mathcal{S}$. Feasibility of Problem 3 then implies each $\widehat{\mathcal{S}}_i$ is nonempty. These assumptions on feasibility are reasonable, since each u_i is unconstrained, and sufficiently large controls can be applied as needed to steer the state into the feasible set.

We also impose the following regularity assumptions: the terminal cost $C(\cdot)$ is L_C-Lipschitz continuous on $\mathcal{S}$, and, for each i, the terminal state estimator $\mathcal{M}_{t_i}^\theta(\cdot)$ is $L_{\mathcal{M}_x,i}$-Lipschitz continuous on the region of interest.

For the same initial state $x_0 = \bar{x}_0$, assume both Problem 2 and Problem 3 are solved to global optimality, and denote their optimal control sequences by u^{P2} and u^{P3}, respectively. Since $\mathcal{J}$ is the objective of Problem 2, by definition we have $0 \le \mathcal{J}(x_0, u^{\text{P3}}) - \mathcal{J}(x_0, u^{\text{P2}})$. It remains to upper-bound this difference.

Remark 4. *The global optimality assumption does not conflict with the fact that Problem 2 is a local optimal control formulation. As discussed in Related Work, local versus global optimal control refers to the scope of control: open-loop for a fixed initial state, versus a feedback policy defined on all states. This is distinct from local versus global optimality of a single optimization problem. A local optimal control problem can still be solved to global optimality; in that case, the trajectory under the resulting open-loop control coincides with that produced by the optimal feedback policy starting from the same state.*

Define the terminal value function $V_N(x) \coloneqq C(x) + \iota_{\mathcal{S}}(x)$, where $\iota_{\mathcal{S}}(x)$ denotes the indicator of $\mathcal{S}$:

$$\iota_{\mathcal{S}}(x) = \begin{cases} 0, & x \in \mathcal{S}, \\ +\infty, & x \notin \mathcal{S}. \end{cases}$$

For $i = 0, 1, \cdots, N-1$, define the optimal cost-to-go as

$$\begin{aligned} V_i(x) \coloneqq \min_{\{x_j\}_{j=i}^N, \{u_j\}_{j=i}^{N-1}} \quad & C(x_N) + \lambda_{\text{oc}} \textstyle\sum_{j=i}^{N-1} \frac{1}{2} \|u_j\|_2^2 \Delta t_j \\ \text{s.t.} \quad & x_i = x, \\ & x_{j+1} = \Psi_j^\theta(x_j, u_j), \\ & j = i, i+1, \cdots, N-1, \\ & h(x_N) \le 0. \end{aligned}$$

For any function $F : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ and $i = 0, 1, \cdots, N-1$, define the Bellman operators

$$(T_i F)(x) := \min_u \left\{ l_i(u) + F\left(\Psi_i^\theta(x,u)\right) \right\},$$

which can be equivalently written as

$$(T_i F)(x) := \min_y \left\{ \frac{\lambda_{\text{oc}}}{2\Delta t_i} \left\| y - \Psi_i^\theta(x) \right\|_2^2 + F(y) \right\}. \quad (13)$$

This expression uses the same control-to-state change of variables as in Problem 4. We have the following proposition.

Proposition 2. *For $i = 0, 1, \cdots, N-1$, the Bellman recursion $V_i = T_i V_{i+1}$ holds, and we have $V_0(x_0) = \mathcal{J}(x_0, u^{\text{P2}})$.*

Proof. The result follows from Bellman's principle of optimality and standard dynamic programming arguments. See, e.g., [50] for details. □

Instead of the true terminal value $V_N(x)$, Problem 3 employs proxy terminal values $\widehat{V}_i(x) \coloneqq C(\mathcal{M}_{t_i}^\theta(x)) + \iota_{\mathcal{S}}(\mathcal{M}_{t_i}^\theta(x))$, for $i = 1, 2, \cdots, N$. By definition of u^{P3}, we have

$$u_i^{\text{P3}} = \operatorname*{argmin}_u \left\{ l_i(u) + \widehat{V}_{i+1}\left(\Psi_i^\theta(x_i, u)\right) \right\},$$

for $i = 0, 1, \cdots, N-1$.

Given $x \in \widehat{\mathcal{S}}_i$, define the Bellman residual

$$r_i(x) \coloneqq (T_i \widehat{V}_{i+1})(x) - \widehat{V}_i(x).$$

We explicitly require $x \in \widehat{\mathcal{S}}_i$ so that $\widehat{V}_i(x)$ is finite. The term $(T_i \widehat{V}_{i+1})(x)$ is always finite for any x, since we can always select $y \in \widehat{\mathcal{S}}_{i+1}$ in (13).

Denote $\left\{x_i^{\text{P3}}\right\}_{i=0}^N$ as the state trajectory under u^{P3} starting from x_0. Since u^{P3} is feasible for Problem 3, it then follows that $x_i^{\text{P3}} \in \widehat{\mathcal{S}}_i$ for $i = 1, 2, \cdots, N$. Evaluating the Bellman residual at these states, we have

$$\begin{aligned} r_i\left(x_i^{\text{P3}}\right) &= T_i \widehat{V}_{i+1}\left(x_i^{\text{P3}}\right) - \widehat{V}_i\left(x_i^{\text{P3}}\right) \\ &= l_i\left(u_i^{\text{P3}}\right) + \widehat{V}_{i+1}\left(x_{i+1}^{\text{P3}}\right) - \widehat{V}_i\left(x_i^{\text{P3}}\right). \end{aligned}$$

Summing over $i = 1, 2, \cdots, N-1$ yields

$$\begin{aligned} \sum_{i=1}^{N-1} r_i\left(x_i^{\text{P3}}\right) &= \sum_{i=1}^{N-1} l_i\left(u_i^{\text{P3}}\right) + \sum_{i=1}^{N-1} \left(\widehat{V}_{i+1}\left(x_{i+1}^{\text{P3}}\right) - \widehat{V}_i\left(x_i^{\text{P3}}\right) \right) \\ &= \sum_{i=1}^{N-1} l_i\left(u_i^{\text{P3}}\right) + \widehat{V}_N\left(x_N^{\text{P3}}\right) - \widehat{V}_1\left(x_1^{\text{P3}}\right) \\ &= \mathcal{J}\left(x_0, u^{\text{P3}}\right) - l_0\left(u_0^{\text{P3}}\right) - \widehat{V}_1\left(x_1^{\text{P3}}\right). \end{aligned}$$

Since $(T_0 V_1)(x_0) = V_0(x_0) = \mathcal{J}(x_0, u^{\text{P2}})$ and $l_0\left(u_0^{\text{P3}}\right) + \widehat{V}_1\left(x_1^{\text{P3}}\right) = \left(T_0 \widehat{V}_1\right)(x_0)$, we have

$$\mathcal{J}(x_0, u^{\text{P3}}) - \mathcal{J}(x_0, u^{\text{P2}}) = \sum_{i=1}^{N-1} r_i\left(x_i^{\text{P3}}\right) + (T_0 \widehat{V}_1 - T_0 V_1)(x_0).$$

To deal with the $(T_0 \widehat{V}_1 - T_0 V_1)(x_0)$ term, we use the following proposition. We assume the minimizer from $T_i V_{i+1}$ lies in $\widehat{\mathcal{S}}_{i+1}$. This effectively requires that the proxy terminal constraints in Problem 3 do not exclude the optimal successor state. If this assumption is violated, the bound can be refined by adding a term that accounts for the resulting minimizer mismatch. The extension is routine but tedious, so we present the simpler form here.

Proposition 3. *For any $i = 0, 1, \cdots, N-1$, any $x \in \mathbb{R}^d$, we have*

$$(T_i \widehat{V}_{i+1} - T_i V_{i+1})(x) \le \sup_{y \in \widehat{\mathcal{S}}_{i+1}} (\widehat{V}_{i+1} - V_{i+1})(y).$$

Proof. Let $y_x \in \widehat{\mathcal{S}}_{i+1}$ attain the minimum in $(T_i V_{i+1})(x) = \frac{\lambda_{\text{oc}}}{2\Delta t_i} \left\| y_x - \Psi_i^\theta(x) \right\|_2^2 + V_{i+1}(y_x)$. Then

$$T_i \widehat{V}_{i+1}(x) \le \frac{\lambda_{\text{oc}}}{2\Delta t_i} \left\| y_x - \Psi_i^\theta(x) \right\|^2 + \widehat{V}_{i+1}(y_x).$$

Therefore,

$$T_i\widehat{V}_{i+1}(x) - T_iV_{i+1}(x) \le \widehat{V}_{i+1}(y_x) - V_{i+1}(y_x) \le \sup_{y\in\widehat{\mathcal{S}}_{i+1}} \left(\widehat{V}_{i+1} - V_{i+1}\right)(y).$$

□

For any $i = 1, 2, \cdots, N$ and $x \in \widehat{\mathcal{S}}_i$, we have

$$\widehat{V}_i(x) - V_i(x) = T_i\widehat{V}_{i+1}(x) - T_iV_{i+1}(x) - r_i(x).$$

Taking supremum over $x \in \widehat{\mathcal{S}}_i$, we obtain

$$\sup_{x\in\widehat{\mathcal{S}}_i}(\widehat{V}_i - V_i)(x) \le \sup_{x\in\widehat{\mathcal{S}}_i}(T_i\widehat{V}_{i+1} - T_iV_{i+1})(x) + \|r_i\|_{\infty,\widehat{\mathcal{S}}_i} \le \sup_{y\in\widehat{\mathcal{S}}_{i+1}} \left(\widehat{V}_{i+1} - V_{i+1}\right)(y) + \|r_i\|_{\infty,\widehat{\mathcal{S}}_i},$$

where the second inequality follows from Proposition 3. We use the notation $\|f\|_{\infty,\mathcal{X}} = \sup_{x\in\mathcal{X}}|f(x)|$ for any function f and set $\mathcal{X}$. Define $E_i \coloneqq \sup_{x\in\widehat{\mathcal{S}}_i}(\widehat{V}_i - V_i)(x)$ for $i = 1, 2, \cdots, N$. Then we have the recursion $E_i \le E_{i+1} + \|r_i\|_{\infty,\widehat{\mathcal{S}}_i}$ for $i = 1, 2, \cdots, N-1$, and $E_N = 0$ since $\widehat{V}_N = V_N$. Unrolling the recursion yields

$$E_1 \le \sum_{i=1}^{N-1} \|r_i\|_{\infty,\widehat{\mathcal{S}}_i}.$$

Then we have the bound

$$(T_0\widehat{V}_1 - T_0V_1)(x_0) \le \sup_{y\in\widehat{\mathcal{S}}_1}(\widehat{V}_1 - V_1)(y) = E_1 \le \sum_{i=1}^{N-1} \|r_i\|_{\infty,\widehat{\mathcal{S}}_i}.$$

Combining the above results, we arrive at

$$\mathcal{J}(x_0, u^{\mathrm{P3}}) - \mathcal{J}(x_0, u^{\mathrm{P2}}) \le \sum_{i=1}^{N-1} \left(\|r_i\|_{\infty,\widehat{\mathcal{S}}_i} + r_i\left(x_i^{\mathrm{P3}}\right)\right) \le 2\sum_{i=1}^{N-1} \|r_i\|_{\infty,\widehat{\mathcal{S}}_i}.$$

To complete the analysis, we bound the Bellman residuals. Assume that for each $i = 0, 1, \ldots, N-1$ there exists $\kappa_i \in (0, \infty)$ such that, for every $x \in \widehat{\mathcal{S}}_i$, there is a feasible control u with $\|u\|_2 \le \kappa_i$ and $\Psi_i^\theta(x, u) \in \widehat{\mathcal{S}}_{i+1}$. This is a reasonable assumption that rules out pathological cases. Expanding the expression for $r_i(x)$ gives

$$r_i(x) = \min_u \left\{l_i(u) + \widehat{V}_{i+1}\left(\Psi_i^\theta(x, u)\right)\right\} - \widehat{V}_i(x).$$

For $x \in \widehat{\mathcal{S}}_i$, pick a feasible u with $\|u\|_2 \le \kappa_i$. We have

$$\begin{aligned} r_i(x) &\le l_i(u) + C\left(\mathcal{M}_{t_{i+1}}^\theta\left(\Psi_i^\theta(x, u)\right)\right) - C\left(\mathcal{M}_{t_i}^\theta(x)\right) \\ &\le \frac{\lambda_{\mathrm{oc}}}{2}\kappa_i^2\Delta t_i + L_C\left(\varepsilon_i + L_{\mathcal{M}_x,i+1}\|u\|\Delta t_i\right) \\ &\le L_C\varepsilon_i + \left(\frac{\lambda_{\mathrm{oc}}}{2}\kappa_i^2 + L_CL_{\mathcal{M}_x,i+1}\kappa_i\right)\Delta t_i, \end{aligned}$$

where

$$\varepsilon_i \coloneqq \sup_{x\in\widehat{\mathcal{S}}_i} \left\|\mathcal{M}_{t_{i+1}}^\theta\left(\Psi_i^\theta(x)\right) - \mathcal{M}_{t_i}^\theta(x)\right\|_2$$

is called the one-step consistency error of the terminal state estimator. For a lower bound, we use $C\left(\mathcal{M}_{t_{i+1}}^\theta\left(\Psi_i^\theta(x, u)\right)\right) \ge C\left(\mathcal{M}_{t_{i+1}}^\theta\left(\Psi_i^\theta(x)\right)\right) - L_CL_{\mathcal{M}_x,i+1}\|u\|\Delta t_i$. Thus,

$$\begin{aligned} r_i(x) &\ge \min_u \left\{\frac{\lambda_{\mathrm{oc}}}{2}\|u\|^2\Delta t_i - L_CL_{\mathcal{M}_x,i+1}\|u\|\Delta t_i\right\} - L_C\varepsilon_i \\ &= -\frac{L_C^2L_{\mathcal{M}_x,i+1}^2}{2\lambda_{\mathrm{oc}}}\Delta t_i - L_C\varepsilon_i. \end{aligned}$$

Combining both sides gives

$$\|r_i\|_{\infty,\widehat{\mathcal{S}}_i} \le L_C\varepsilon_i + \Gamma_i\Delta t_i,$$

where

$$\Gamma_i \coloneqq \max\left\{\frac{L_C^2L_{\mathcal{M}_x,i+1}^2}{2\lambda_{\mathrm{oc}}}, \frac{\lambda_{\mathrm{oc}}}{2}\kappa_i^2 + L_CL_{\mathcal{M}_x,i+1}\kappa_i\right\}.$$

Therefore, we have established the following theorem.

Theorem 1. *For any initial state $x_0 = \bar{x}_0 \sim p_0$, if both Problem 2 and Problem 3 are solved to global optimality, whose solutions are denoted as u^{P2} and u^{P3}, then*

$$0 \le \mathcal{J}\left(x_0, u^{\mathrm{P3}}\right) - \mathcal{J}\left(x_0, u^{\mathrm{P2}}\right) \le 2\sum_{i=1}^{N-1}\left(L_C\varepsilon_i + \Gamma_i\Delta t_i\right). \tag{14}$$

Proof. The result follows by combining $\mathcal{J}\left(x_0, u^{\mathrm{P3}}\right) - \mathcal{J}\left(x_0, u^{\mathrm{P2}}\right) \le 2\sum_{i=1}^{N-1}\|r_i\|_{\infty,\widehat{\mathcal{S}}_i}$ with the previous analysis on $\|r_i\|_{\infty,\widehat{\mathcal{S}}_i}$. □

Remark 5. *The bound in Theorem 1 directly reflects the two sources of approximation errors when transforming Problem 2 to Problem 3. First, the one-step consistency errors ε_i arise from replacing the true terminal state obtained by ODE simulation with the estimator $\mathcal{M}_{t_i}^\theta(\cdot)$. With a perfect terminal-state oracle, ε_i would vanish to zero. Second, when computing u_i, we ignore the future controls $\{u_j\}_{j=i+1}^{N-1}$, which introduces the term $\frac{L_C^2L_{\mathcal{M}_x,i+1}^2}{2\lambda_{\mathrm{oc}}}$ in Γ_i. The remaining term $\frac{\lambda_{\mathrm{oc}}}{2}\kappa_i^2 + L_CL_{\mathcal{M}_x,i+1}\kappa_i$ accounts for the control effort needed to steer the state into the feasible set when required.*

Remark 6. *Since Problem 2 converges to Problem 1 as $N \to \infty$, it is natural to ask what happens to Theorem 1 in the continuous-time limit. The bound* (14) *remains well-behaved as $N \to \infty$. It suffices to show that the one-step consistency errors satisfy $\varepsilon_i = \mathcal{O}(\Delta t_i)$. Assume in addition that $t \mapsto \mathcal{M}_t^\theta(x)$ is $L_{\mathcal{M}_t,i}$-Lipschitz on the region of interest (uniformly in x), and that the velocity is bounded, $V_i \coloneqq \sup_{x\in\widehat{\mathcal{S}}_i}\left\|v_{t_i}^\theta(x)\right\| < \infty$. Then*

$$\begin{aligned} \varepsilon_i &= \sup_{x\in\widehat{\mathcal{S}}_i} \left\|\mathcal{M}_{t_{i+1}}^\theta\left(\Psi_i^\theta(x)\right) - \mathcal{M}_{t_i}^\theta(x)\right\|_2 \\ &\le \sup_{x\in\widehat{\mathcal{S}}_i} \left\|\mathcal{M}_{t_{i+1}}^\theta\left(\Psi_i^\theta(x)\right) - \mathcal{M}_{t_{i+1}}^\theta(x)\right\|_2 \\ &\quad + \sup_{x\in\widehat{\mathcal{S}}_i} \left\|\mathcal{M}_{t_{i+1}}^\theta(x) - \mathcal{M}_{t_i}^\theta(x)\right\|_2 \\ &\le \left(L_{\mathcal{M}_x,i+1}V_i + L_{\mathcal{M}_t,i}\right)\Delta t_i = \mathcal{O}\left(\Delta t_i\right). \end{aligned}$$

Next, for completeness, we state the equivalence of Problem 3, Problem 4, and Problem 5 in the following theorem.

Theorem 2. *For any initial state $x_0 = \bar{x}_0$, running Problem 3, Problem 4, or Problem 5 produces the same control trajectory $\{u_i\}_{i=0}^{N-1}$ and state trajectory $\{x_i\}_{i=0}^{N}$.*

Proof. The equivalence follows directly from the definitions and straightforward algebraic manipulations. □

Lastly, we analyze the gap between Problem 5 and Problem 6, which is introduced by the one-step fixed-point iteration. Both problems share the same decision variable $\widehat{x}_N$ and the same constraints $h(\widehat{x}_N) \leq 0$. The difference lies in the quadratic penalty term in the objective. Problem 5 uses $\|(\mathcal{M}^{\theta}_{t_{i+1}})^{-1}(\widehat{x}_N) - \bar{x}_{i+1}\|_2^2$, whereas Problem 6 uses $\|\mathcal{F}^{\theta}_{t_{i+1}}(\widehat{x}_N) - \bar{x}_{i+1}\|_2^2$. Assume $\mathcal{N}^{\theta}_{t_i}(\cdot)$ is $L_{\mathcal{N}_x,i}$-Lipschitz continuous on the region of interest. Assume $\left|\beta_{t_{i+1}}\right| L_{\mathcal{N}_x,i+1} < 1$ for $i = 0, 1, \cdots, N-1$. We quantify the objective difference in the following theorem.

Theorem 3. *For $i = 0, 1, \cdots, N-1$ and any $y \in \mathbb{R}^d$, denote the subproblem objective in Problem 5 as $\mathcal{J}_i^{\mathrm{P5}}(y) \coloneqq C(y) + \frac{\lambda_{\mathrm{oc}}}{2\Delta t_i}\|(\mathcal{M}^{\theta}_{t_{i+1}})^{-1}(y) - \bar{x}_{i+1}\|_2^2$, and that in Problem 6 as $\mathcal{J}_i^{\mathrm{P6}}(y) \coloneqq C(y) + \frac{\lambda_{\mathrm{oc}}}{2\Delta t_i}\|\mathcal{F}^{\theta}_{t_{i+1}}(y) - \bar{x}_{i+1}\|_2^2$. Then, we have*

$$\left|\mathcal{J}_i^{\mathrm{P5}}(y) - \mathcal{J}_i^{\mathrm{P6}}(y)\right| \leq \frac{\lambda_{\mathrm{oc}}}{2\Delta t_i}\frac{r(2-r)}{(1-r)^2}\alpha_{t_{i+1}}^2\|y - \bar{y}\|^2, \quad (15)$$

where $r = \left|\beta_{t_{i+1}}\right| L_{\mathcal{N}_x,i+1}$ and $\bar{y} = \mathcal{M}^{\theta}_{t_{i+1}}(\bar{x}_{i+1})$.

Proof. $C(y)$ cancels, so only the quadratic terms differ. Recall (7) and (8). Set $x^*(y) = \left(\mathcal{M}^{\theta}_{t_{i+1}}\right)^{-1}(y)$ and $x^{(0)} = \bar{x}_{i+1}$. Taking one-step fixed-point iteration, we have $x^{(1)}(y) = T^y_{t_{i+1}}(x^{(0)}) = \alpha_{t_{i+1}}y + \beta_{t_{i+1}}\mathcal{N}^{\theta}_{t_{i+1}}(\bar{x}_{i+1}) = \mathcal{F}^{\theta}_{t_{i+1}}(y)$. Since by assumption $\left|\beta_{t_{i+1}}\right| L_{\mathcal{N}_x,i+1} < 1$, the operator $T^y_{t_{i+1}}(\cdot)$ is a contraction with factor $r = \left|\beta_{t_{i+1}}\right| L_{\mathcal{N}_x,i+1}$. Hence

$$\begin{aligned}
&\|x^{(1)}(y) - x^*(y)\|_2 = \left\|T^y_{t_{i+1}}\left(x^{(0)}\right) - T^y_{t_{i+1}}\left(x^*\right)\right\|_2 \\
&\leq r\|x^{(0)} - x^*(y)\|_2 \\
&\leq r\left(\left\|x^{(0)} - x^{(1)}(y)\right\|_2 + \left\|x^{(1)}(y) - x^*(y)\right\|_2\right).
\end{aligned}$$

Therefore,

$$\|x^{(1)}(y) - x^*(y)\|_2 \leq \frac{r}{1-r}\|x^{(0)} - x^{(1)}(y)\|_2.$$

Using (3) at $\bar{x}_{i+1}$, we have $x^{(1)}(y) - x^{(0)} = \alpha_{t_{i+1}}(y - \bar{y})$, where $\bar{y} = \mathcal{M}^{\theta}_{t_{i+1}}(\bar{x}_{i+1})$. Let $a = x^*(y) - x^{(0)}$, $b = x^{(1)}(y) - x^{(0)}$, and $c = b - a = x^{(1)}(y) - x^*(y)$. Then $\left|\|a\|_2^2 - \|b\|_2^2\right| = \left|\|b - c\|_2^2 - \|b\|_2^2\right| = \left|-2\langle b, c\rangle + \|c\|_2^2\right| \leq 2\|b\|_2\|c\|_2 + \|c\|_2^2$. With $\|b\|_2 = \left|\alpha_{t_{i+1}}\right|\|y - \bar{y}\|_2$ and $\|c\|_2 \leq \frac{r}{1-r}\|b\|_2$, we obtain

$$\begin{aligned}
\left|\|a\|^2 - \|b\|_2^2\right| &\leq \alpha_{t_{i+1}}^2\|y - \bar{y}\|_2^2\left(\frac{2r}{1-r} + \frac{r^2}{(1-r)^2}\right) \\
&= \alpha_{t_{i+1}}^2\|y - \bar{y}\|_2^2\frac{2r - r^2}{(1-r)^2}.
\end{aligned}$$

Multiplying by $\frac{\lambda_{\mathrm{oc}}}{2\Delta t_i}$ yields (15) as claimed. □

Remark 7. *Theorem 3 requires the quantity $\left|\beta_{t_{i+1}}\right| L_{\mathcal{N}_x,i+1}$ to be small. For affine conditional paths, the boundary condition $\beta_1 = 0$ makes this requirement easier to satisfy as the sampling process approaches the final time $t = 1$. This is also consistent with Theorem 1, where the suboptimality gap depends on one-step consistency errors governed by the accuracy of terminal state estimators $\mathcal{M}^{\theta}_{t_i}(\cdot)$, which typically improve over time. These observations suggest a practical heuristic: skip control in the early stages of sampling, and activate the subproblems in later steps, when both the estimator and the fixed-point approximation are more reliable.*

VII. Experiments

In this section, we evaluate our algorithm HardFlow on a diverse set of tasks from various domains, namely, robotic manipulation, maze navigation, partial differential equation (PDE) control, and text-guided image editing. Results across these distinct domains demonstrate the generality and effectiveness of our proposed framework.

In every task, a pretrained flow-matching model $v_t^{\theta}(x)$ is provided. There are hard constraints $h(x) \leq 0$ (see Remark 1), which may be complex and involve multiple equalities and inequalities. There are also costs $C(x)$ capturing additional preferences over samples. Our goal is to generate samples that satisfy the hard constraints $h(x) \leq 0$ while minimizing the costs $C(x)$. We will describe $v_t^{\theta}(x)$, $h(x)$, and $C(x)$ in detail for each task later.

For comparison, we implement state-of-the-art baselines from the two families discussed in Related Work: soft-constrained and hard-constrained approaches. For the soft-constrained family, we add penalties for constraint violations to the original cost $C(x)$ and then apply training-free guidance to minimize the augmented cost during sampling. Specifically, we implement **OC-Flow** [25], which perturbs the sampling trajectory via indirect optimal control, and a standard gradient-based guidance approach (hereafter **Gradient Guidance**) used widely in prior work [43]–[45], which updates samples using the cost evaluated at the posterior mean. For the hard-constrained family, the central strategy is projection during sampling, with most prior work focused on diffusion models. We consider three representative projection-based methods and adapt them to flow-matching models: (i) **Projection-All** [15], [16], [20], [21], projecting after every sampling step; (ii) **Projection-Late** [17], [22], projecting only in the later sampling steps; and (iii) **Projection-Relaxed** [18], [19], which performs a fixed number of augmented Lagrangian iterations at each sampling step, thereby relaxing constraints in early sampling steps when the multipliers are small. Since projection-based methods only handle constraint enforcement, we also report enhanced versions that combine them with Gradient Guidance on the cost $C(x)$ during sampling, enabling a fair comparison with our approach, which addresses cost minimization and constraint satisfaction in a unified framework. Lastly, we include standard sampling without any guidance, denoted as **Original**, to demonstrate the capabilities of the base model.

All experiments are conducted on a desktop with an Intel Core Ultra 9 285K CPU and an NVIDIA GeForce RTX 5090 GPU. For all tasks, we use the scheduler $(\alpha_t = t, \beta_t = 1 - t)$ and Euler discretization with uniform timesteps, which is a simple and widely used choice in the flow-matching literature. Additional experiment details are reported in the Appendix.

A. Robotic Manipulation

This task follows the setup of [16] and uses the D3IL benchmark introduced in [51]. A robotic manipulator aims to reach a target region with its end-effector while avoiding obstacles, as shown in Fig. 2. The state $s \in \mathbb{R}^4$ comprises the current and desired two-dimensional positions of the end-effector. The action $a \in \mathbb{R}^2$ is the desired two-dimensional velocity of the end-effector. D3IL provides expert demonstrations that weave around six pillars to reach a target region. We train a flow-matching model to generate trajectories of horizon H, i.e., samples $x = (s_0, a_0, s_1, a_1, \cdots, s_{H-1}, a_{H-1}) \in \mathbb{R}^{6H}$. At test time, conditioned on the current state s, we sample a trajectory with $s_0 = s$ and execute the actions $\{a_i\}_{i=0}^{T-1}$ sequentially in the environment, where $T \leq H$ denotes the replanning horizon. We also introduce novel test-time obstacles that conflict with many trajectories in the training data, thus increasing task difficulty. The hard constraints $h(x) \leq 0$ ensure that the trajectory avoids all obstacles, i.e., both the pillars and the purple regions. Since a sample x is a trajectory of the robotic system, we also incorporate dynamics constraints into $h(x) \leq 0$ to ensure physical fidelity [16], [22]: $s_{i+1} = f(s_i, a_i)$ for $i = 0, 1, \cdots, H-1$, where f is fitted from the training data. The cost $C(x)$ is the squared distance between the final state s_{H-1} and the target region, which encourages reaching the target as fast as possible. The optimization problem (11) in HardFlow, as well as the projection operations in the baselines, is solved using the open-source nonlinear programming solver IPOPT [52].

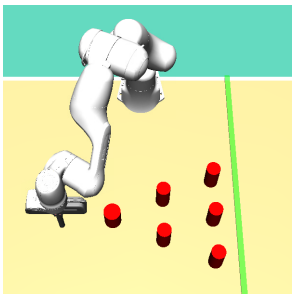

Fig. 2: The robotic manipulation task.

We evaluate all methods over 50 trials. The simulation starts at the same initial position and terminates when the end-effector reaches the target area or collides with any obstacle. A visualization of our algorithm is shown in Fig. 3. We record three metrics: (i) Safety Rate: the percentage of trials without collision; (ii) Total Steps (Safe Trials): the average number of steps that the robot takes to reach the target among trials not terminated by collision; and (iii) Computation Time: the average time taken to sample a trajectory from the flow-matching model at each replanning step. The quantitative results are summarized in Table I. Our algorithm HardFlow is the only method to achieve a perfect safety rate (1.00), while also requiring the fewest steps to reach the target and incurring only mild computational overhead. In contrast, other baselines exhibit much lower safety rates and longer paths.

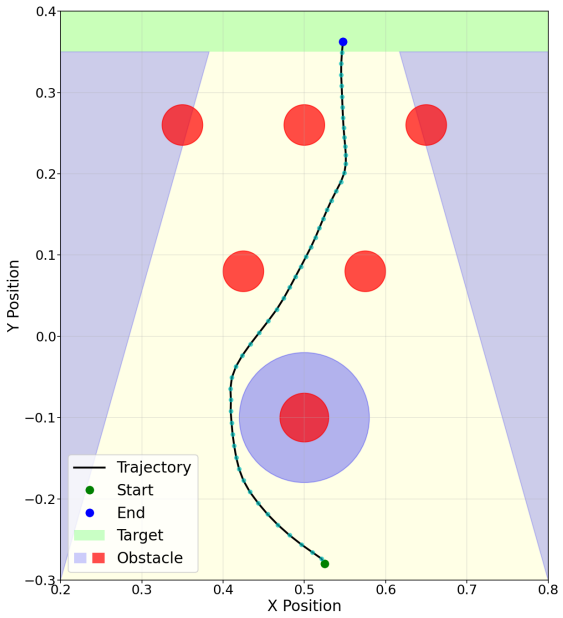

Fig. 3: Visualization of HardFlow in the robotic manipulation task. The green region denotes the target area, the red circles represent original obstacles, and the purple region denotes the novel obstacles introduced at test time. The robot successfully weaves around all obstacles to reach the target.

B. Maze Navigation

This task follows the setup of [18], [19], [53], which are based on the Maze2D environment from D4RL [54]. As shown in Fig. 4, a force-actuated ball navigates a two-dimensional maze to a goal position. The state $s \in \mathbb{R}^4$ consists of the two-dimensional position and velocity of the ball. The action $a \in \mathbb{R}^2$ is the applied two-dimensional force. We train a flow-matching model to generate trajectories of horizon H, i.e., samples $x = (s_0, a_0, s_1, a_1, \cdots, s_{H-1}, a_{H-1}) \in \mathbb{R}^{6H}$. At test time, a trajectory is sampled conditioned on the initial state s_0 and the goal state s_{H-1}. A proportional-derivative (PD) controller is deployed to track the positions in this trajectory. The hard constraints $h(x) \leq 0$ enforce obstacle avoidance for the two red regions in Fig. 4. As in the robotic manipulation task, we impose dynamics constraints for physical fidelity: $s_{i+1} = f(s_i, a_i)$ for $i = 0, 1, \cdots, H-1$, where f is fitted from the training data. We use IPOPT to solve the optimization problem (11) and the projection steps in the baselines. The cost $C(x)$ is the (squared) total length of the trajectory, encouraging fast progress to the goal.

We evaluate all methods over 50 trials. A visualization of our algorithm is shown in Fig. 4. We report four metrics: (i) Safety Rate: the percentage of trials without entering the red obstacles; (ii) Violations: the average number of timesteps during which the ball is inside the obstacles; (iii) Score: the D4RL normalized score [54] (faster goal reaching yields higher scores); and (iv) Computation Time: the average time taken to sample a trajectory from the flow-matching model. The quantitative results are summarized in Table II. Our method HardFlow is the only approach to achieve a perfect safety rate (1.00) with zero violations while also obtaining the

TABLE I: Results on the robotic manipulation task. Values of the form $a \pm b$ indicate mean $\pm$ standard deviation. HardFlow is the only method to achieve a perfect safety rate (1.00), while also requiring the fewest steps to reach the target and incurring only mild computational overhead. In contrast, other baselines exhibit much lower safety rates and longer paths.

Method	Safety Rate	Total Steps (Safe Trials)	Computation Time (s)
Original	0.06	58.7 $\pm$ 4.0	0.060 $\pm$ 0.009
Gradient Guidance	0.18	61.7 $\pm$ 6.3	0.992 $\pm$ 0.022
OC-Flow	0.14	63.9 $\pm$ 6.3	0.847 $\pm$ 0.016
Projection-All	0.46	67.2 $\pm$ 7.1	0.349 $\pm$ 0.051
Projection-Late	0.76	67 $\pm$ 11	0.236 $\pm$ 0.072
Projection-Relaxed	0.10	63.4 $\pm$ 7.9	0.116 $\pm$ 0.014
Projection-All + Gradient Guidance	0.40	70.6 $\pm$ 8.3	1.556 $\pm$ 0.057
Projection-Late + Gradient Guidance	0.68	67.2 $\pm$ 9.3	1.380 $\pm$ 0.035
Projection-Relaxed + Gradient Guidance	0.04	65.0 $\pm$ 2.8	1.074 $\pm$ 0.016
HardFlow (ours)	**1.00**	**52.5 $\pm$ 4.4**	0.190 $\pm$ 0.023

TABLE II: Results on the maze navigation task. Values of the form $a \pm b$ indicate mean $\pm$ standard deviation. HardFlow is the only approach to achieve a perfect safety rate (1.00) with zero violations while also obtaining the highest score. Its computation time is comparable to or lower than most baselines.

Method	Safety Rate	Violations	Score	Computation Time (s)
Original	0.02	49 $\pm$ 20	1.47 $\pm$ 0.45	0.082 $\pm$ 0.034
Gradient Guidance	0.88	1.4 $\pm$ 4.2	1.11 $\pm$ 0.75	6.666 $\pm$ 0.076
OC-Flow	0.04	35 $\pm$ 17	1.47 $\pm$ 0.45	3.983 $\pm$ 0.085
Projection-All	0.22	38 $\pm$ 30	1.41 $\pm$ 0.54	10.4 $\pm$ 1.6
Projection-Late	0.30	34 $\pm$ 31	1.51 $\pm$ 0.39	6.2 $\pm$ 1.2
Projection-Relaxed	0.00	33 $\pm$ 14	1.53 $\pm$ 0.32	4.11 $\pm$ 0.33
Projection-All + Gradient Guidance	0.24	37 $\pm$ 30	1.56 $\pm$ 0.34	25.7 $\pm$ 3.1
Projection-Late + Gradient Guidance	0.30	34 $\pm$ 30	1.50 $\pm$ 0.39	18.02 $\pm$ 0.74
Projection-Relaxed + Gradient Guidance	0.00	25 $\pm$ 10	1.53 $\pm$ 0.32	11.20 $\pm$ 0.19
HardFlow (ours)	**1.00**	**0.0 $\pm$ 0.0**	**1.620 $\pm$ 0.010**	4.09 $\pm$ 0.82

highest score. Its computation time is comparable to or lower than most baselines.

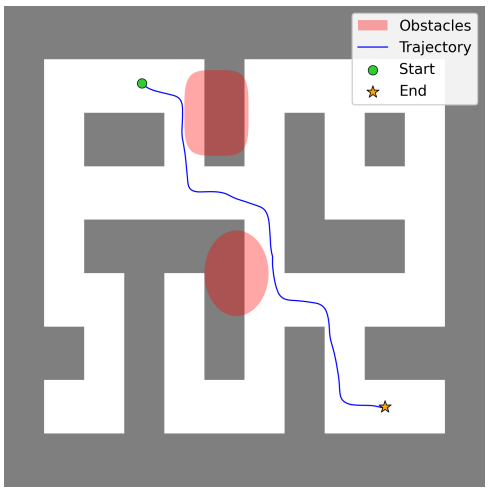

Fig. 4: Visualization of HardFlow in the maze navigation task. The red region denotes the introduced obstacles. The robot successfully navigates from the start position to the goal position while avoiding obstacles.

C. PDE Control

One-dimensional Burgers' equation is a fundamental PDE that models various physical phenomena, such as fluid dynamics and traffic flow. Following [55], we consider Dirichlet boundary conditions, with state $u(t, s)$ and external control $f(t, s)$:

$$\begin{cases} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial s} = \nu\frac{\partial^2 u}{\partial s^2} + f(t,s) & (t,s) \in [0,T] \times [0,L] \\ u(0,s) = u_0(s), u(T,s) = u_T(s) & s \in [0,L] \\ u(t,0) = u(t,L) = 0 & t \in [0,T] \end{cases} \tag{16}$$

We train a flow-matching model to generate solutions to PDE (16). Each sample x contains the discretized states and controls on a spatiotemporal grid with m time steps and n spatial points. The dataset is from [55], containing PDE solutions under diverse initial and terminal conditions. The hard constraints $h(x) \leq 0$ enforce time-varying bounds on $|u|$. For physical fidelity, we also enforce a discretized form of (16) on each sample x, where the viscosity parameter ν is treated as uncertain within a known interval, i.e, $\nu \in [\nu_{\min}, \nu_{\max}]$. The cost $C(x)$ is the total control energy. We solve (11) in HardFlow and the projection steps in the baselines with IPOPT.

Conditioned on given boundary conditions $u(0, s) = u_0(s)$ and $u(T, s) = u_T(s)$, we sample from the flow-matching model, extract the predicted control f, and simulate the PDE on a fine grid to obtain the resulting controlled state u. We evaluate all methods over 50 trials, each with different boundary conditions. A visualization of controlled state u and predicted control f at one trial is shown in Fig. 5. State slices at three time instants are presented in Fig. 6, demonstrating that the controlled state complies with the time-varying bounds. Following [55], we report three metrics quantifying constraint satisfaction of the state u: (i) $\mathcal{R}_{\text{sample}}$: the fraction

TABLE III: Results on the PDE control task. $\mathcal{R}_{\text{sample}}$ denotes the fraction of trials with any violation; $\mathcal{R}_{\text{time}}$ denotes the fraction of unsafe timesteps over all timesteps; and $\mathcal{R}_{\text{point}}$ denotes the fraction of spatial points that ever violate a constraint across all timesteps. Values of the form $a \pm b$ indicate mean $\pm$ standard deviation. HardFlow is one of four methods that achieve perfect constraint satisfaction (zero on all three safety metrics). Among them, HardFlow attains the lowest control energy and the shortest computation time.

Method	$\mathcal{R}_{\text{sample}}$	$\mathcal{R}_{\text{time}}$	$\mathcal{R}_{\text{point}}$	**Control Energy**	**Computation Time (s)**
Original	1.00	0.77	0.48	0.59 ± 0.26	0.162 ± 0.085
Gradient Guidance	0.36	0.13	0.04	0.31 ± 0.10	2.71 ± 0.15
OC-Flow	0.78	0.47	0.19	0.180 ± 0.075	4.434 ± 0.034
Projection-All	**0.00**	**0.00**	**0.00**	0.53 ± 0.16	13.6 ± 1.2
Projection-Late	**0.00**	**0.00**	**0.00**	0.51 ± 0.16	8.9 ± 2.7
Projection-Relaxed	1.00	0.72	0.42	0.49 ± 0.21	0.20 ± 0.17
Projection-All + Gradient Guidance	**0.00**	**0.00**	**0.00**	0.36 ± 0.12	17.2 ± 1.3
Projection-Late + Gradient Guidance	**0.00**	**0.00**	**0.00**	0.36 ± 0.12	11.5 ± 2.5
Projection-Relaxed + Gradient Guidance	0.94	0.64	0.30	0.27 ± 0.12	2.84 ± 0.20
HardFlow (ours)	**0.00**	**0.00**	**0.00**	$\mathbf{0.28 \pm 0.10}$	8.3 ± 1.1

of trials with any violation; (ii) $\mathcal{R}_{\text{time}}$: the fraction of unsafe timesteps over all timesteps; and (iii) $\mathcal{R}_{\text{point}}$: the fraction of spatial points that ever violate a constraint across all timesteps. We also report Control Energy and Computation Time. The quantitative results are summarized in Table III. HardFlow is one of four methods that achieve perfect constraint satisfaction (zero on all three safety metrics). Among them, HardFlow attains the lowest control energy and the shortest computation time.

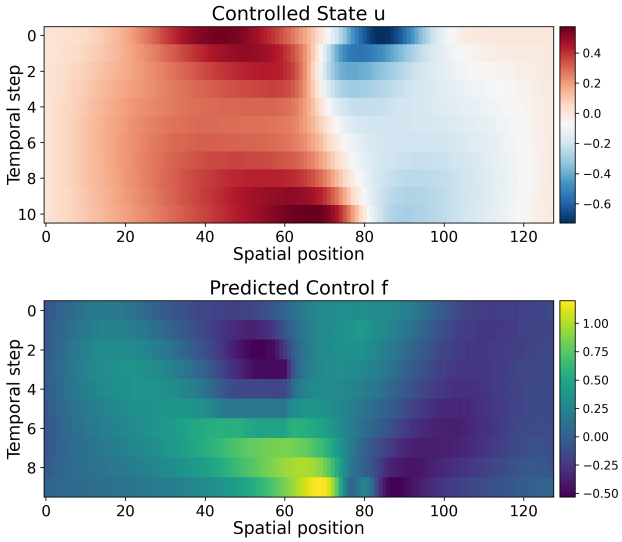

Fig. 5: Heatmaps of controlled state u and predicted control f produced by HardFlow at one trial in the PDE control task. The grid uses $m = 10$ time steps and $n = 128$ spatial points; the axis ticks reflect this discretization.

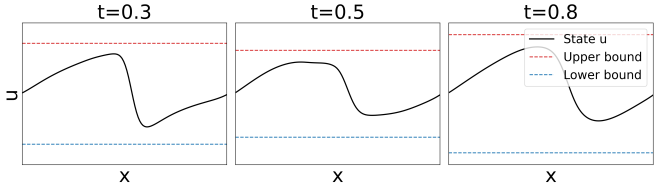

Fig. 6: Slices of controlled state u at three time instants, produced by HardFlow. The state constraints are satisfied.

D. Text-Guided Image Editing

In this task, leveraging a flow-matching model pretrained on the CelebA-HQ celebrity face dataset [56], we edit an input image according to a text prompt so that the edited image aligns with the prompt requirements. We follow the setup of [25]. The pretrained flow-matching model is from [57]. Given an input image, we integrate the flow ODE backward from $t = 1$ to $t = 0$ to recover the corresponding initial noise x_0, then apply a guided sampling method to generate the edited image x_1. We use CLIP [58] to score how well an image matches the text prompt and define the cost $C(x)$ as the negative CLIP score. However, a potential risk is that the edits are so substantial that the original identity of the input image is lost. To address this issue, we use LPIPS [59] to quantify perceptual similarity between the edited and original images, and impose a hard constraint $h(x) \leq 0$ requiring LPIPS to remain below an upper bound. We test with five text prompts: "A photo of an angry face.", "A photo of a face with curly hair.", "A photo of an old face.", "A photo of a sad face.", and "A photo of a smiling face.", which are referred to as Angry, Curly, Old, Sad, and Smile hereafter.

We evaluate on 200 images randomly sampled from the CelebA-HQ validation set. In this image editing setting, the Original baseline is not meaningful, as it simply returns the input image unchanged. The constraints-only baselines (Projection-All/Late/Relaxed) are also inapplicable: without optimizing the cost (i.e., CLIP score), there is no incentive to modify the image, so the LPIPS constraint is trivially satisfied. In addition, since the LPIPS constraint is computed by a neural network and the sampling uses many discretization steps ($N = 100$), exact projection at every step or many steps, as in Projection-All/Late, is computationally prohibitive. Therefore, we compare HardFlow with Gradient Guidance, OC-Flow, and Projection-Relaxed + Gradient Guidance. The optimization problem (11) is solved with a fixed number of augmented Lagrangian iterations. Quantitative results by prompt are shown in Fig. 7, and aggregated results are summarized in Table IV. Safety Rate represents the fraction of edited images satisfying the LPIPS constraint. HardFlow is the only method to achieve 100% constraint satisfaction across all prompts and images. OC-Flow and Gradient Guidance

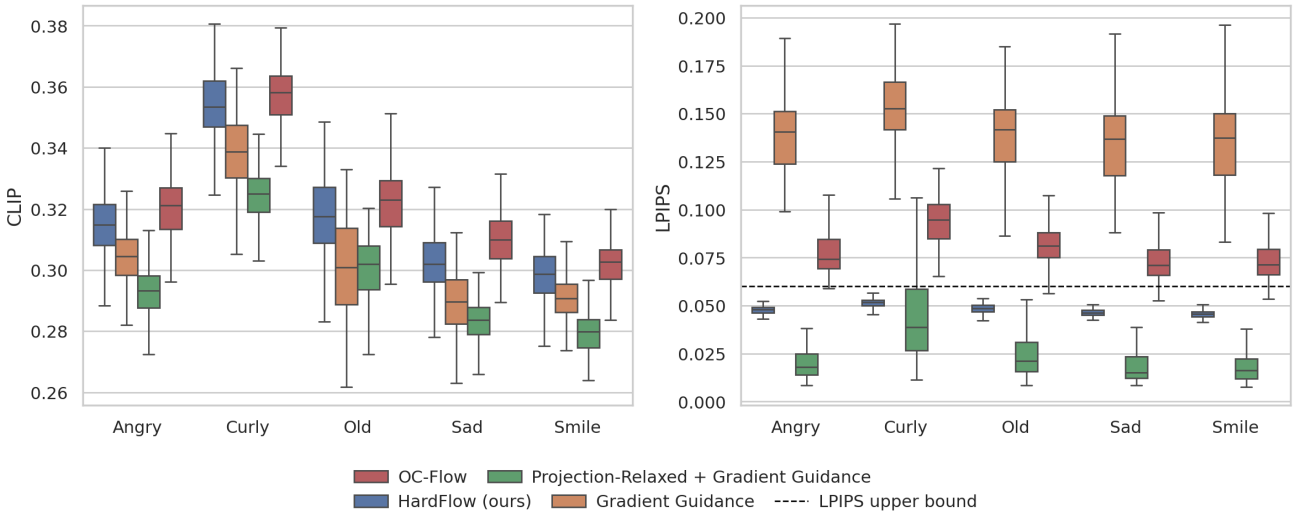

Fig. 7: Box-and-whisker plots of CLIP (left; higher is better) and LPIPS (right; below the dashed line is required) for five prompts on the text-guided image editing task. For each prompt and method, the box spans the interquartile range (25th to 75th percentiles), the horizontal line marks the median, and the whiskers extend to 1.5 times the interquartile range. The dashed line in the LPIPS panel indicates the LPIPS upper bound.

TABLE IV: Results on the text-guided image editing task. CLIP measures editing effectiveness (higher is better). LPIPS measures perceptual similarity between the edited and original images, and is required to be below 0.06. Safety rate denotes the percentage of edits that satisfy the constraint. HardFlow is the only method to achieve 100% constraint satisfaction across all prompts and images, attains strong CLIP scores, and is substantially faster than all baselines.

Method	**Safety Rate**	**LPIPS**	**CLIP**	**Computation Time (s)**
OC-Flow	0.033	0.082 $\pm$ 0.009	0.322 $\pm$ 0.021	158.302 $\pm$ 0.016
Gradient Guidance	0.000	0.144 $\pm$ 0.007	0.304 $\pm$ 0.019	131.819 $\pm$ 0.076
Projection-Relaxed + Gradient Guidance	0.939	0.026 $\pm$ 0.011	0.296 $\pm$ 0.018	129.336 $\pm$ 0.047
HardFlow (ours)	**1.000**	**0.048** $\pm$ **0.002**	**0.317** $\pm$ **0.022**	51.294 $\pm$ 0.004

substantially violate the LPIPS constraint, indicating over-editing that alters identity. Projection-Relaxed + Gradient Guidance drives LPIPS unnecessarily low, even though the requirement is only to stay below the upper bound, which in turn hurts its CLIP score. HardFlow achieves the second-best CLIP score, very close to OC-Flow, and is much more computationally efficient than all baselines.

To further illustrate the benefits of imposing hard constraints on LPIPS, we provide qualitative comparisons in Fig. 8. The three reference images are shown in the top-left of Fig. 8, with their CelebA-HQ numbers displayed beneath each. We refer to them by these numbers hereafter. A notable issue with OC-Flow is that it appears to change the perceived gender of the subject, for example, in the Angry, Curly, and Sad edits of 002140, and the Curly and Smile edits of 014314. Both OC-Flow and Gradient Guidance can alter facial features excessively, deviating too much from the original identity, for example, the Angry and Smile edits of 004666 by OC-Flow and most edits of 004666 and 014314 by Gradient Guidance. Gradient Guidance also changes hair color drastically in all edits of 002140, which is unnecessary for the given prompts. Projection-Relaxed + Gradient Guidance yields much worse visual quality despite decent CLIP and LPIPS values, suggesting reward hacking in which the numerical scores appear acceptable even though the outputs are visually degraded. This may stem from constraining intermediate states rather than the final state in the sampling process.

VIII. Conclusion

In this paper, we addressed the critical challenge of hard-constrained sampling for flow-matching models. We proposed a method that departs from common projection-based techniques by reformulating the sampling process as a trajectory optimization problem. The central idea is to utilize numerical optimal control to guide the generation trajectory so that constraints are met precisely at the final step, thereby avoiding the unnecessary restrictions and quality degradation associated with constraining the entire path. By leveraging the intrinsic structure of flow-matching models in conjunction with techniques from model predictive control, we obtain a tractable and efficient algorithm for an otherwise complex control problem. This control-theoretic perspective offers significant flexibility beyond mere feasibility, enabling the integration of auxiliary objectives that minimize distribution shift and enhance sample quality within a unified framework. The soundness of our approach is supported by theoretical analysis, and its practical efficacy and robustness are confirmed through extensive experiments across diverse domains. Empirically, our technique

Fig. 8: Qualitative comparison of different methods on the text-guided image editing task. The three reference images are shown in the top-left, with their CelebA-HQ numbers displayed beneath each. Below each edited image, CLIP and LPIPS are shown. LPIPS values that satisfy the constraint are displayed in green, and violations are displayed in red.

consistently and substantially outperforms existing methods, delivering superior results in both constraint satisfaction and sample quality. This work opens promising new avenues for applying trajectory optimization techniques, particularly direct methods, to generative models, presenting a powerful and flexible paradigm for controllable generation.

Acknowledgment

We sincerely thank Wei-Cheng Huang for insightful discussions on optimal control and trajectory optimization.

Appendix A
Experiment Details

A. Robotic Manipulation

The flow-matching model is a temporal U-Net following [53], trained on the D3IL dataset [51] with training details as in [45]. The trajectory horizon is $H = 16$ and the replanning horizon is $T = 8$. Obstacle-avoidance constraints for the red pillars and purple regions follow [16]. The dynamics constraints are $s_{i+1} = As_i + Ba_i + c$ for $i = 0, \dots, H-2$, where $A \in \mathbb{R}^{4\times4}$, $B \in \mathbb{R}^{4\times2}$, and $c \in \mathbb{R}^4$ are fitted via least squares on the training data. The cost $C(\cdot)$ is the squared distance from s_{H-1} to the target region. During sampling, we use $N = 10$ discretization steps for all methods. OC-Flow is implemented following [25]. Given a cost function $\widehat{C}(\cdot)$, at sampling step i Gradient Guidance updates the sample x using the gradient $\nabla_x \widehat{C}(x + (1-t_i)\, v^\theta_{t_i}(x))$. We use $\widehat{C}(\cdot) = C(\cdot) + C_{\text{penalty}}(\cdot)$ for Gradient Guidance and OC-Flow, where $C_{\text{penalty}}(\cdot)$ applies quadratic penalties to constraint violations. In Projection-All, we project at every sampling step; in Projection-Late, we project during the second half of the steps; and in Projection-Relaxed, the augmented Lagrangian implementation follows [19]. When combining projection-based methods with Gradient Guidance, the guidance cost is the original $C(\cdot)$ (without penalties), since constraint enforcement is handled by projection. In HardFlow, we solve (11) only during the second half of the sampling steps, since the posterior estimates are less accurate early in sampling (see Remark 7). All projection operations in related baselines and the optimization problem (11) in HardFlow are solved using IPOPT with default settings.

B. Maze Navigation

The trajectory horizon is $H = 384$. Obstacle-avoidance constraints for the red regions follow [18]. The cost $C(x)$ is the (squared) total length of the trajectory. The flow-matching

model is trained on the D4RL maze2d-large-v1 dataset [54]. All other implementation details are identical to those in the robotic manipulation task.

C. PDE Control

The flow-matching model is a temporal U-Net, with architecture and dataset following [55]. The temporal domain is $t \in [0,1]$ and the spatial domain is $s \in [0,1]$ (i.e., $T = L = 1$). The spatiotemporal grid uses $m = 10$ time steps and $n = 128$ spatial points. The time-varying state constraints are defined as $|u(t,s)| \leq 0.8 \cdot (2t^2 - 2t + 1)$. The dynamics constraints come from a finite-difference discretization of the Burgers' PDE, with viscosity ν treated as uncertain within $[\nu_{\min}, \nu_{\max}] = [0.0, 0.02]$. The cost $C(\cdot)$ is the total control energy, i.e., the sum of squared control values over all (t,s) grid points. All other implementation details are identical to those in the robotic manipulation task.

D. Text-Guided Image Editing

We adopt the full pipeline of [25], including the flow-matching model, the CLIP and LPIPS models, and image preprocessing. Let $f(x) = \text{CLIP}(x)$ and $g(x) = \text{LPIPS}(x, x_{\text{ref}})$, where x_{ref} is the input image. The cost is $C(x) = -f(x)$. The constraint is $g(x) \leq \epsilon$, where $\epsilon = 0.06$ denotes the LPIPS upper bound. During sampling, we use $N = 100$ discretization steps for all methods. For HardFlow, we solve (11) with an augmented Lagrangian method. At sampling step i, the augmented Lagrangian is $\mathcal{L}_i(x, \lambda, \rho) = -f(x) + f_{\text{reg}}(x) - \lambda(g(x) - \epsilon) - \frac{\rho}{2}(\max\{0, g(x) - \epsilon\})^2$, where $f_{\text{reg}}(x)$ denotes the regularization term corresponding to $\alpha_{t_{i+1}}^2 \left\| \widehat{x}_N - \bar{x}_N \right\|_2^2$ in (11). Since posterior estimates are less accurate in early steps (see Remark 7), we omit the LPIPS constraint during the first half of the sampling steps (i.e., forcing $\mathcal{L}_i = -f(x) + f_{\text{reg}}(x)$ for $i < N/2$). We run 40 gradient-ascent steps on $\mathcal{L}_i$ per sampling step i with the same learning rate $\eta = 2.5$ as OC-Flow, during which λ and ρ are updated every 8 steps. For OC-Flow and Gradient Guidance, we use the guidance cost $\widehat{C}(x) = -f(x) + (\max\{0, g(x) - \epsilon\})^2$, which encodes the LPIPS constraint as a fixed-weight penalty. For Projection-Relaxed + Gradient Guidance, we follow [19] to enforce the LPIPS constraint $g(x) \leq \epsilon$, while using $-f(x)$ as the guidance cost.

REFERENCES

[1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. pmlr, 2015, pp. 2256–2265.

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[3] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *The Eleventh International Conference on Learning Representations*, 2023.

[4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[5] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *Forty-first international conference on machine learning*, 2024.

[6] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," *Advances in neural information processing systems*, vol. 35, pp. 8633–8646, 2022.

[7] Y. Jin, Z. Sun, N. Li, K. Xu, K. Xu, H. Jiang, N. Zhuang, Q. Huang, Y. Song, Y. MU, and Z. Lin, "Pyramidal flow matching for efficient video generative modeling," in *The Thirteenth International Conference on Learning Representations*, 2025.

[8] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles *et al.*, "De novo design of protein structure and function with rfdiffusion," *Nature*, vol. 620, no. 7976, pp. 1089–1100, 2023.

[9] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[10] H. Ding, N. Jaquier, J. Peters, and L. Rozo, "Fast and robust visuomotor riemannian flow matching policy," *IEEE Transactions on robotics*, 2025.

[11] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[12] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[13] Z. Zhang, S. Zhang, L. Shen, Y. Zhan, Y. Luo, H. Hu, B. Du, Y. Wen, and D. Tao, "Aligning text-to-image diffusion models with constrained reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[14] T. Power, R. Soltani-Zarrin, S. Iba, and D. Berenson, "Sampling constrained trajectories using composable diffusion models," in *IROS 2023 Workshop on Differentiable Probabilistic Robotics: Emerging Perspectives on Robot Learning*, 2023.

[15] J. K. Christopher, S. Baek, and N. Fioretto, "Constrained synthesis with projected diffusion models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 89 307–89 333, 2024.

[16] R. Römer, A. v. Rohr, and A. Schoellig, "Diffusion predictive control with constraints," in *Proceedings of the 7th Annual Learning for Dynamics & Control Conference*, ser. Proceedings of Machine Learning Research, N. Ozay, L. Balzano, D. Panagou, and A. Abate, Eds., vol. 283. PMLR, 04–06 Jun 2025, pp. 791–803.

[17] Y. Yuan, J. Song, U. Iqbal, A. Vahdat, and J. Kautz, "Physdiff: Physics-guided human motion diffusion model," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 16 010–16 021.

[18] W. Xiao, T.-H. Wang, C. Gan, R. Hasani, M. Lechner, and D. Rus, "Safediffuser: Safe planning with diffusion probabilistic models," in *The Thirteenth International Conference on Learning Representations*, 2025.

[19] J. Zhang, L. Zhao, A. Papachristodoulou, and J. Umenberger, "Constrained diffusers for safe planning and control," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[20] H. Luan, Y. X. Goh, S.-K. Ng, and C. K. Ling, "Projected coupled diffusion for test-time constrained joint generation," *arXiv preprint arXiv:2508.10531*, 2025.

[21] S. Zampini, J. K. Christopher, L. Oneto, D. Anguita, and F. Fioretto, "Training-free constrained generation with stable diffusion models," *arXiv preprint arXiv:2502.05625*, 2025.

[22] J.-B. Bouvier, K. Ryu, Q. Liao, K. Sreenath, and N. Mehr, "DDAT: Diffusion Policies Enforcing Dynamically Admissible Robot Trajectories," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025.

[23] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *The Ninth International Conference on Learning Representations*, 2021.

[24] L. Rout, Y. Chen, N. Ruiz, A. Kumar, C. Caramanis, S. Shakkottai, and W.-S. Chu, "Rb-modulation: Training-free stylization using reference-based modulation," in *The Thirteenth International Conference on Learning Representations*, 2025.

[25] L. Wang, C. Cheng, Y. Liao, Y. Qu, and G. Liu, "Training free guided flow-matching with optimal control," in *The Thirteenth International Conference on Learning Representations*, 2025.

[26] D. E. Kirk, *Optimal Control Theory: An Introduction*. Courier Corporation, 2004.

[27] O. Von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of operations research*, vol. 37, no. 1, pp. 357–373, 1992.

[28] B. A. Conway, "A survey of methods available for the numerical optimization of continuous dynamic systems," *Journal of Optimization Theory and Applications*, vol. 152, no. 2, pp. 271–306, 2012.

[29] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & chemical engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[30] K. Mizuta and K. 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)*. IEEE, 2024, pp. 13 801–13 808.

[31] J. Carvalho, A. T. Le, P. Kicki, D. Koert, and J. Peters, "Motion planning diffusion: Learning and adapting robot motion planning with diffusion models," *IEEE Transactions on Robotics*, 2025.

[32] D. J. Bell and D. H. Jacobson, *Singular Optimal Control Problems*. Elsevier, 1975, vol. 117.

[33] W. H. Fleming and R. W. Rishel, *Deterministic and Stochastic Optimal Control*. Springer Science & Business Media, 2012, vol. 1.

[34] B. Tzen and M. Raginsky, "Theoretical guarantees for sampling and inference in generative models with latent diffusions," in *Conference on Learning Theory*. PMLR, 2019, pp. 3084–3114.

[35] Q. Zhang and Y. Chen, "Path integral sampler: A stochastic control approach for sampling," in *The Tenth International Conference on Learning Representations*, 2022.

[36] F. Vargas, W. S. Grathwohl, and A. Doucet, "Denoising diffusion samplers," in *The Eleventh International Conference on Learning Representations*, 2023.

[37] J. Berner, L. Richter, and K. Ullrich, "An optimal control perspective on diffusion-based generative modeling," *Transactions on Machine Learning Research*, 2024.

[38] A. J. Havens, B. K. Miller, B. Yan, C. Domingo-Enrich, A. Sriram, D. S. Levine, B. M. Wood, B. Hu, B. Amos, B. Karrer, X. Fu, G.-H. Liu, and R. T. Q. Chen, "Adjoint sampling: Highly scalable diffusion samplers via adjoint matching," in *Forty-second International Conference on Machine Learning*, 2025.

[39] C. Domingo-Enrich, M. Drozdzal, B. Karrer, and R. T. Chen, "Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control," in *The Thirteenth International Conference on Learning Representations*, 2025.

[40] H. Zhao, H. Chen, J. Zhang, D. Yao, and W. Tang, "Score as action: Fine tuning diffusion generative models by continuous-time reinforcement learning," in *Forty-second International Conference on Machine Learning*, 2025.

[41] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye, "Diffusion posterior sampling for general noisy inverse problems," in *The Eleventh International Conference on Learning Representations*, 2023.

[42] A. Bansal, H.-M. Chu, A. Schwarzschild, S. Sengupta, M. Goldblum, J. Geiping, and T. Goldstein, "Universal guidance for diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 843–852.

[43] Y. Guo, H. Yuan, Y. Yang, M. Chen, and M. Wang, "Gradient guidance for diffusion models: An optimization perspective," *Advances in Neural Information Processing Systems*, vol. 37, pp. 90 736–90 770, 2024.

[44] K. Pandey, F. M. Sofian, F. Draxler, T. Karaletsos, and S. Mandt, "Variational control for guidance in diffusion models," in *Forty-second International Conference on Machine Learning*, 2025.

[45] R. Feng, C. Yu, W. Deng, P. Hu, and T. Wu, "On the guidance of flow matching," in *Forty-second International Conference on Machine Learning*, 2025.

[46] Y. Huang, A. Ghatare, Y. Liu, Z. Hu, Q. Zhang, C. S. Sastry, S. Gururani, S. Oore, and Y. Yue, "Symbolic music generation with non-differentiable rule guided diffusion," in *International Conference on Machine Learning*. PMLR, 2024, pp. 19 772–19 797.

[47] V. Jain, K. Sareen, M. Pedramfar, and S. Ravanbakhsh, "Diffusion tree sampling: Scalable inference-time alignment of diffusion models," *arXiv preprint arXiv:2506.20701*, 2025.

[48] Y. Guo, Y. Yang, H. Yuan, and M. Wang, "Training-free guidance beyond differentiability: Scalable path steering with tree search in diffusion and flow models," *arXiv preprint arXiv:2502.11420*, 2025.

[49] Y. Wang, J. Yu, and J. Zhang, "Zero-shot image restoration using denoising diffusion null-space model," in *The Eleventh International Conference on Learning Representations*, 2023.

[50] D. Bertsekas, *Dynamic Programming and Optimal Control: Volume I*. Athena Scientific, 2012, vol. 4.

[51] X. Jia, D. Blessing, X. Jiang, M. Reuss, A. Donat, R. Lioutikov, and G. Neumann, "Towards diverse behaviors: A benchmark for imitation learning with human demonstrations," in *The Twelfth International Conference on Learning Representations*, 2024.

[52] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[53] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162. PMLR, 2022, pp. 9902–9915.

[54] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[55] P. Hu, X. Qian, W. Deng, R. Wang, H. Feng, R. Feng, T. Zhang, L. Wei, Y. Wang, Z.-M. Ma, and T. Wu, "From uncertain to safe: Conformal adaptation of diffusion models for safe PDE control," in *Forty-second International Conference on Machine Learning*, 2025.

[56] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *The Sixth International Conference on Learning Representations*, 2018.

[57] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *The Eleventh International Conference on Learning Representations*, 2023.

[58] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[59] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.