# INTEGRATION MATTERS: ROLLOUT-BASED TRAINING FOR CONSTRAINED DIFFUSION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Constrained generative models aim to produce samples that satisfy complex feasibility constraints while remaining faithful to the underlying data distribution. Existing methods typically enforce constraints through either training-time optimization or sampling-time correction. Training-time approaches optimize over states induced by the training distribution, which can differ substantially from those encountered during sampling. Sampling-time correction methods instead modify the sampling process at inference, introducing distribution shift and requiring costly tuning, particularly in few-step sampling regimes. We propose IMCD (Integration Matters for Constrained Diffusion), a fine-tuning framework that integrates constraint guidance into training through online rollouts. IMCD aligns training with sampling by differentiating through the fixed noise schedule used to numerically integrate the denoising process. This exposes the model to constraint violations that arise along the denoising trajectory and aligns diffusion learning with the sampling process. Experiments across multiple tasks show that our method improves constraint satisfaction while maintaining competitive sampling quality compared to prior methods.

## 1 INTRODUCTION

In many real-world applications, generative tasks are governed not only by the underlying data distribution but also by complex feasibility constraints that determine whether generated samples are valid. In robotics systems, generated trajectories must remain geometrically feasible and avoid collisions with the environment and other agents (Schulman et al., 2014; Carvalho et al., 2023). Similarly, in autonomous driving scenarios, predicted trajectories must remain on the road and obey traffic rules (Yang et al., 2024a; Zheng et al., 2025). Diffusion models (Ho et al., 2020; Song et al., 2021; Karras et al., 2022) provide a powerful framework for high-quality generation (Rombach et al., 2022; Ho et al., 2022; Harvey et al., 2022). Despite their strong generative performance, reliably enforcing feasibility constraints in diffusion models remains challenging.

This challenge stems in part from a discrepancy between diffusion training and sampling. During training, denoising predictions are optimized independently at randomly selected noise levels, whereas sampling sequentially integrates these predictions along a complete denoising trajectory. As a result, prediction errors can accumulate along the integration trajectory and produce terminal constraint violations that the training objective does not directly penalize. Existing constrained diffusion models address this limitation through either constraint-aware fine-tuning (Bastek et al., 2024; Naderiparizi et al., 2025; Liang et al., 2026) or inference-time guidance (Dhariwal & Nichol, 2021; Chung et al., 2022; He et al., 2024; Yu et al., 2023; Lou & Ermon, 2023; Fishman et al., 2023; Ye et al., 2024; Christopher et al., 2024; Liang et al., 2025). Fine-tuning methods often evaluate constraint objectives on forward noising states rather than denoising rollout states, while inference-time methods modify the sampling trajectory of the pretrained model and require additional guidance tuning (Chung et al., 2024; Yang et al., 2024b). Both approaches therefore leave a mismatch between constraint optimization and the sampling process.

We tackle constrained generation with a fine-tuning approach that aligns training with sampling. Our key insight is that **integration matters**: rather than optimizing constraints on forward-noised states or enforcing constraints through sampling-time corrections, we optimize the model through online denoising rollouts that match the specific integration scheme used at test time. Constraint violations



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

are measured on the terminal generated samples, and the resulting objective trains the model on the states it encounters during sampling. To preserve distributional fidelity, we retain the EDM denoising loss (Karras et al., 2022) as a regularizer.

Our contributions are as follows: (1) We show that constraint supervision vanishes on forward-noised states but persists along the reverse denoising trajectory, and introduce a fine-tuning framework that optimizes constraint satisfaction through the reverse denoising trajectory. (2) We provide a formal analysis of the training-time alignment induced by our training objective and show that the terminal denoising state converges towards the feasible region. (3) We validate our method on two constrained generation tasks: bouncing ball and traffic scene trajectory prediction experiments.

## 2 BACKGROUND

**Diffusion model generation** Diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021) generate samples by gradually corrupting data with noise and then learning to reverse this process through denoising. Following EDM (Karras et al., 2022), we perturb data $\mathbf{x}_0 \sim p_0$ as $\mathbf{x}_t = \mathbf{x}_0 + t\epsilon$, with $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and $t = \sigma(t)$ directly denotes the noise level. This corresponds to the stochastic differential equation (SDE) $d\mathbf{x}_t = g(t)\, d\mathbf{w}_t$, whose reverse-time SDE is given by (Anderson, 1982):

$$d\mathbf{x}_t = -2t\, \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\, dt + \sqrt{2t}\, d\bar{\mathbf{w}}_t. \tag{1}$$

where $\bar{\mathbf{w}}_t$ denotes a Wiener process evolving backward in time. Since the score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is intractable, diffusion models learn a neural approximation $s_\theta(\mathbf{x}_t; t)$ trained via the denoising score matching (DSM) objective (Vincent, 2011). Through Tweedie's formula (Efron, 2011), score estimation can be recast as a denoising problem. Given a denoiser network $D_\theta(\mathbf{x}_t; t)$ that approximates $\mathbb{E}\left[\mathbf{x}_0 \mid \mathbf{x}_t\right]$, the score can be estimated as

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \left(\mathbb{E}\left[\mathbf{x}_0 \mid \mathbf{x}_t\right] - \mathbf{x}_t\right)/t^2 \approx \left(D_\theta(\mathbf{x}_t; t) - \mathbf{x}_t\right)/t^2. \tag{2}$$

**Constrained diffusion generation** In many applications, generated samples are required to satisfy problem-specific constraints that define a feasible set $\Omega \subseteq \mathbb{R}^d$. These constraints often do not admit an explicit analytic boundary and are only implicitly specified through task-specific functions. We assume access to a differentiable constraint violation function $\ell^\Omega(\mathbf{x})$, where $\ell^\Omega(\mathbf{x}) = 0$ if $\mathbf{x} \in \Omega$ and $\ell^\Omega(\mathbf{x}) > 0$ otherwise. We further assume that the data distribution is supported on the feasible set, $\mathrm{supp}(p_0) \subseteq \Omega$, although $\Omega$ may contain points outside $\mathrm{supp}(p_0)$. Accordingly, a positive value of $\ell^\Omega(\mathbf{x})$ quantifies the degree of constraint violation, whereas a zero value indicates only feasibility, and does not imply that $\mathbf{x}$ lies in the support of $p_0$. Constrained diffusion generation seeks to produce samples through a reverse process that remains faithful to $p_0$ while satisfying the feasibility condition of lying in $\Omega$.

However, learning such constraints from data alone is challenging, as training data only consists of feasible examples and lacks explicit information about infeasible regions (Regenwetter et al., 2024). Prior work (Naderiparizi et al., 2024) further shows that increasing dataset size alone does not eliminate constraint violations, leaving the model's behavior outside the data support and instead determined by inductive biases. Consequently, maximum-likelihood training alone cannot achieve high constraint satisfaction, motivating the need for additional signals that explicitly penalize infeasible generations.

## 3 LIMITATIONS OF EXISTING APPROACHES TO CONSTRAINED DIFFUSION

### 3.1 TRAINING-TIME MISALIGNMENT IN CONSTRAINT FINE-TUNING

Prior constrained diffusion methods (Bastek et al., 2024; Liang et al., 2026) derive their training signals from the constraint function evaluated at $D_\theta(\mathbf{x}_t; t)$ for forward-noised states $\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0)$, where $\mathbf{x}_0$ is a feasible training sample. Since $\Omega$ is defined over clean samples, we evaluate constraint violations on the denoised estimate $D_\theta(\mathbf{x}_t; t)$ rather than on the noisy state $\mathbf{x}_t$ itself. At low noise levels, $\mathbf{x}_t$ remains close to $\mathbf{x}_0$, and the pretrained denoiser typically reconstructs an estimate near the underlying feasible data point (Biroli et al., 2024). Consequently, $\ell^\Omega(D_\theta(\mathbf{x}_t; t)) \approx 0$, providing

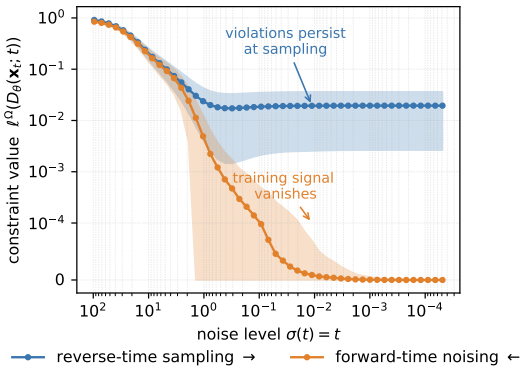
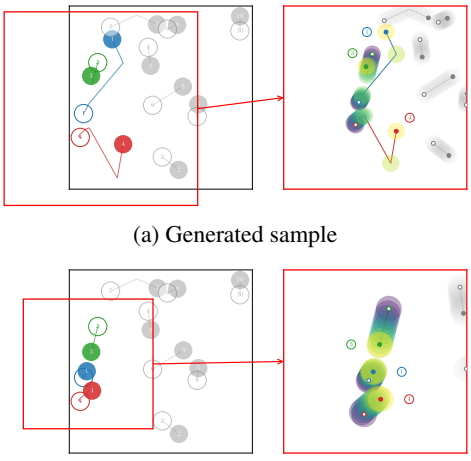



Figure 1: Constraint violation $\ell^{\Omega}(D_\theta(\mathbf{x}_t; t))$ of the denoised prediction across noise levels $t$ in the bouncing ball experiment. The intermediates states $\mathbf{x}_t$ are obtained either by forward-noising feasible data samples or through reverse-time sampling. Here, $\ell^{\Omega}$ combines the boundary and overlap violations. Curves show the mean, and shaded regions indicate standard deviations.



(a) Generated sample

(b) Ground truth sample

Figure 2: Generated versus ground-truth bouncing ball trajectories.

little supervision for learning constraint correction. This observation applies specifically to low-noise inputs obtained by forward-noising feasible data. In contrast, sampling starts from Gaussian noise and numerically integrates a sequence of discrete denoising steps, allowing model approximation and discretization errors to accumulate and persist to low noise levels. The resulting infeasible rollout states require correction but are rarely encountered under the forward-noising distribution used for training. This leads to a misalignment: the model is trained mostly on near-feasible states, yet must correct infeasible rollout states at inference.

This training-sampling misalignment is illustrated in Figure 1 using the bouncing-ball constraints from the experiment in Section 5.1. In this setting, the constraint violation function $\ell^{\Omega}(\cdot)$ is the sum of two components: the boundary violation Equation (62), which quantifies ball-wall penetration, and the overlap violation Equation (63), which quantifies ball-ball intersections. At each matched noise level, we evaluate $\ell^{\Omega}(D_\theta(\mathbf{x}_t; t))$ on the denoised prediction, with $\mathbf{x}_t$ obtained either by forward-noising a feasible data sample or from a reverse sampling trajectory of the same EDM model. While both exhibit similar behavior at high noise levels, they diverge significantly at low noise. Forward states remain near-feasible and yield near-zero constraint loss, whereas reverse states incur persistent violations. This indicates that training on forward states underestimates the violations encountered during sampling.

Prior work has studied the discrepancy between forward sampling states $\mathbf{x}_t$ and reverse sampling states $\tilde{\mathbf{x}}_t$ (Ning et al., 2023; Li et al., 2023). We examine its effect by comparing constraint losses on the denoised predictions $D_\theta(\mathbf{x}_t; t)$ and $D_\theta(\tilde{\mathbf{x}}_t; t)$. This shows that constraint signals remain weak on forward sampled states, but become nontrivial along the reverse denoising trajectory.

### 3.2 Sampling-time correction: distribution shift and few-step degradation

Another common approach enforces constraints only at sampling time, using either projection (Christopher et al., 2024; Liang et al., 2025) or gradient-based updates (He et al., 2024)

$$d\mathbf{x}_t = \left[f(\mathbf{x}_t; t) - g^2(t)(s_\theta(\mathbf{x}_t; t) - \gamma(t)\nabla\ell^{\Omega}(\mathbf{x}_t; t))\right] dt + g(t)\, d\bar{\mathbf{w}}_t, \tag{3}$$

where $\gamma(t)$ controls the guidance strength. These methods correct violations online during denoising and can reduce the infeasibility rate in practice, but they are highly sensitive to the guidance schedule (Chung et al., 2022). Weak guidance may fail to remove violations, while strong guidance can push intermediate state $\mathbf{x}_t$ away from the data manifold (Dhariwal & Nichol, 2021), which places samples in regions where the learned score is unreliable. This further widens the misalignment

between forward sampled and reverse sampling states. Moreover, because sampling-time correction relies on incremental corrections across the denoising trajectory, it becomes sensitive to sampling budget: many steps allow small corrections to accumulate, while few steps require stronger updates that can distort or destabilize sampling (Shen et al., 2024).

Figure 2 illustrates this behavior using trajectories generated by a sampling-time correction method called MPGD without projection (He et al., 2024) under the 50-step sampling setting used later in Section 5.1. The full-scene and zoomed views use hollow and solid circles denoting start and end positions. With gray balls shown for context, balls 1, 3, and 4 are highlighted because of their jittering motion, which produces sharp trajectory changes that avoid collision but distort the motion. In contrast, the ground truth rollout remains smooth and physically consistent. Thus, this method enforces feasibility without preserving realistic motion.

These observations reveal two limitations. Training-time optimization suffers from misalignment between the states encountered from training and sampling distributions. Sampling-time correction introduces distribution shift and becomes fragile under few-step sampling. Together, these limitations motivate training on reverse-time rollout states and learning constraint-aware updates that enforce feasibility.

# 4 METHODOLOGY

We present IMCD (Integration Matters for Constrained Diffusion), a rollout-based fine-tuning framework for constrained generative modeling that addresses the training-time misalignment. IMCD evaluates constraint losses at terminal rollout samples and backpropagates through the full denoising trajectory for model updates, aligning optimization with inference. During each rollout step, it computes constraint-aware corrections using denoised lookahead estimates and applies them through a learnable scaling mechanism, enabling adaptive guidance across noise levels. To preserve data fidelity, IMCD retains the EDM loss and updates only lightweight trainable components.

We assume access to a pretrained denoiser denoted by $D$, where the pretrained model is parameterized by fixed model weights $\theta$. Since $\theta$ remains frozen throughout fine-tuning, we omit it from the notation. The fine-tuned denoiser network is initialized from $\theta$ and denoted by $D_\phi$, where $\phi$ collects the lightweight trainable components introduced during fine-tuning. The score functions induced by these denoisers are denoted by $s(\mathbf{x}_t; t)$ and $s_\phi(\mathbf{x}_t; t)$, respectively.

## 4.1 CONSTRAINT GUIDANCE MECHANISM

A key design choice in constrained diffusion is how to construct a constraint-aware score estimator. Generalizing the definitions in (Naderiparizi et al., 2025), we consider estimators of the form

$$s^\Omega_\phi(\mathbf{x}_t; t) = s_\phi(\mathbf{x}_t; t) - \gamma(\mathbf{x}_t; t) G^\Omega(\mathbf{x}_t; t), \tag{4}$$

where $-G^\Omega$ produces a direction that reduces violations of the constraint $\Omega$ and $\gamma$ controls its strength. We call $G^\Omega$ the guidance function and $\gamma$ the scaling function. Table 3 summarizes the definition of these functions in our method and in prior work.

**Guidance function** We follow Liang et al. (2026) for the choice of guidance function $G^\Omega(\mathbf{x}_t; t) := \partial_1 \ell^\Omega(D(\mathbf{x}_t; t); t)$ where $\partial_1$ denotes the partial derivative with respect to the first argument. Effectively, this computes the gradient of the constraint loss function at the one-step denoising prediction by a pretrained denoiser $D$. Liang et al. (2026) show that this leads to a more stable model (both during training and sampling) than computing the gradients at the noisy state $\mathbf{x}_t$.

It is important to note that $\partial_1 \ell^\Omega(D(\mathbf{x}_t; t); t)$ differs from $\nabla_{\mathbf{x}_t} \ell^\Omega(D(\mathbf{x}_t; t); t)$. The latter corresponds more directly to the diffusion formulation, since it fully accounts for the dependence of the denoised prediction on the noisy latent $\mathbf{x}_t$ by backpropagating through the denoiser $D$. In contrast, $\partial_1 \ell^\Omega(D(\mathbf{x}_t; t); t)$ ignores this dependency and does not backpropagate gradients through $D$, making it significantly more computationally efficient in practice.

**Scaling function** The scaling $\gamma$ used in prior work (Naderiparizi et al., 2025; Liang et al., 2026) is independent of the sample $\mathbf{x}_t$ and is typically selected via grid search. In practice, this tuning procedure

can be computationally expensive, as the optimal value of $\gamma$ depends on the sampling regime, noise level, and constraint strength. We instead propose to learn a scaling function that adapts the correction dynamically to both the current state $\mathbf{x}_t$ and diffusion time $t$. This eliminates the need for costly hyperparameter sweeps. This parameterization is also motivated by the exact gradient $\nabla_{\mathbf{x}_t}\ell^\Omega(D(\mathbf{x}_t;t);t)$. In particular, the discrepancy between $\partial_1\ell^\Omega(D(\mathbf{x}_t;t);t)$ and the exact gradient induced by backpropagation through the denoiser can be interpreted as a state- and time-dependent rescaling of the guidance direction. More precisely, $\nabla_{\mathbf{x}_t}\ell^\Omega(D(\mathbf{x}_t;t)) = [\partial D(\mathbf{x}_t;t)/\partial\mathbf{x}_t]^\top \partial_1\ell^\Omega(D(\mathbf{x}_t;t);t)$. Prior work (Efron, 2011; Boys et al., 2023) shows that the Jacobian scales with a posterior covariance term of the form $\partial D(\mathbf{x}_t;t)/\partial\mathbf{x}_t \approx \mathrm{cov}\left[\mathbf{x}_0 \mid \mathbf{x}_t\right]/t^2$. The covariance term captures state-dependent uncertainty, while the denominator amplifies corrections as the sample approaches the data manifold.

**Definition 4.1.** *(Learned scaling network) Given a trainable score $s_\phi$, a constraint violation function $\ell^\Omega(\mathbf{x}_t;t)$, and a learned scaling network $\gamma_\phi : \mathbb{R}^d \times [0,T] \to \mathbb{R}^+$, we define:*

$$\gamma_\phi(\mathbf{x}_t;t) = \alpha_\phi(\mathbf{x}_t;t)\cdot t^{\beta_\phi(\mathbf{x}_t;t)}, \quad \alpha_\phi(\mathbf{x}_t;t) \geq 0, \beta_\phi < -1. \tag{5}$$

*The constrained score is then*

$$s^\Omega_\phi = s_\phi(\mathbf{x}_t;t) - \gamma_\phi(\mathbf{x}_t;t)\partial_1\ell^\Omega(D(\mathbf{x}_t;t);t). \tag{6}$$

This retains the power-law scaling of power work, summarized in Table 3, while learning both its coefficient and exponent as functions of $\mathbf{x}_t$ and $t$.

**Proposition 4.2.** *(Convergence property) Equation* (6) *corresponds to the score of*

$$p^\Omega(\mathbf{x}_t;t) \propto p(\mathbf{x}_t;t)\exp(-\gamma_\phi(\mathbf{x}_t;t)\ell^\Omega(D(\mathbf{x}_t;t))). \tag{7}$$

*Furthermore, $\gamma_\phi(\mathbf{x}_t;t) \to \infty$ as $t \to 0$ for any $\mathbf{x}_t$ with $\alpha_\phi(\mathbf{x}_t;t) > 0$, such that*

$$\lim_{t\to 0} p^\Omega(\mathbf{x}_t;t) \propto \lim_{t\to 0} p(\mathbf{x}_t;t)\mathbf{1}_\Omega(\mathbf{x}_t), \tag{8}$$

*preserving stronger corrections near the data manifold while allowing the magnitude to adapt to the current state and noise level.*

### 4.2 Rollout-based training

With the constraint guidance mechanism in place, we can now close the training-sampling misalignment discussed in Section 3.1. We construct training states by rolling out the reverse-time process from pure Gaussian noise. The training-time rollout shares the same noise sampling schedule as inference such that the intermediate states are generated by the fine-tuned network $D_\phi$.

We use $(\mathbf{x}_t;t)$ for continuous-time notation and adopt step-indexed notation $(\mathbf{x}_i,t_i)$ for the discretized reverse-time rollout. Let the noise schedule be $T = t_N > t_{N-1} > \cdots > t_0 = t_{\min} > 0$. Starting from $\tilde{\mathbf{x}}_N \sim \mathcal{N}(0, t_N^2\boldsymbol{I})$, we simulate denoising steps backward along this schedule and denote the resulting reverse rollout trajectory by $\{\tilde{\mathbf{x}}_i\}_{i=0}^N$. At each rollout step $i$, we compute the denoised state guidance $\partial_1\ell^\Omega(D(\tilde{\mathbf{x}}_i;t_i);t_i)$ using the pretrained denoiser. The fine-tuned denoiser $D_\phi$ is conditioned on this guidance through a lightweight embedding, and the scaling network $\gamma_\phi(\tilde{\mathbf{x}}_i;t_i)$ defined in Equation (5) determines its strength at the current rollout state. This gives the constraint-aware denoiser:

$$D^\Omega_\phi(\tilde{\mathbf{x}}_i;t_i) = D_\phi\left(\tilde{\mathbf{x}}_i;t_i,\partial_1\ell^\Omega(D(\tilde{\mathbf{x}}_i;t_i);t_i)\right) - \gamma_\phi(\tilde{\mathbf{x}}_i;t_i)\partial_1\ell^\Omega(D(\tilde{\mathbf{x}}_i;t_i);t_i). \tag{9}$$

We use this constraint-aware denoiser $D^\Omega_\phi$ at each rollout step. Thus the model receives the constraint signal at its input while also applying an output-level correction with a learned, state-dependent strength. Training on the resulting rollout states $\{\tilde{\mathbf{x}}_i\}$ under the same discretization schedule as inference exposes the model to off-manifold states produced by its own sampling process. This alignment directly targets the training-time misalignment identified in Section 3.1, enabling the model to correct constraint violations during generation.

### 4.3 Training objective and sampling procedure

We optimize the fine-tuning parameters $\phi$ with an EDM denoising term and a terminal rollout constraint term, where $\kappa$ balances constraint optimization against data fidelity:

$$\mathcal{L}(\phi) = \mathcal{L}_{\mathrm{EDM}}(\phi) + \kappa\mathcal{L}_{\mathrm{rollout}}(\phi). \tag{10}$$

**EDM denoising loss** To preserve data-consistent behavior, we retain the EDM denoising objective for the constraint-augmented denoiser $D_\phi^\Omega$:

$$\mathcal{L}_{\mathrm{EDM}}(\phi) = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_t}\left[\lambda(t)\|D_\phi^\Omega(\mathbf{x}_t;t) - \mathbf{x}_0\|^2\right], \tag{11}$$

where $\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0)$ is sampled from the forward process independently of the rollout, and $\lambda(t)$ is the EDM loss weighting. This objective regularizes the model toward data-consistent denoising while allowing deviations required to satisfy the constraints. A KL anchor to the pretrained model explicitly penalizes the corrections needed to remove its own violations in Proposition C.1, so we anchor to the data via the EDM loss instead; Table 5 confirms this empirically.

**Rollout constraint loss** For constraint optimization, constraint loss is applied at the terminal rollout:

$$\mathcal{L}_{\mathrm{rollout}}(\phi) = \mathbb{E}_{p(\tilde{\mathbf{x}}_N),p_\phi(\tilde{\mathbf{x}}_0|\tilde{\mathbf{x}}_N)}\left[\ell^\Omega(\tilde{\mathbf{x}}_0)\right]. \tag{12}$$

During optimization, the sampled states $\{\tilde{\mathbf{x}}_i\}$ are generated by the current denoiser $D_\phi^\Omega$ and are not optimized directly. Gradients of $\ell^\Omega(\tilde{\mathbf{x}}_0)$ are backpropagated through the full rollout to update only the trainable components in $\phi$.

Since supervision is applied only at the terminal state, the constraint signal is sparse and becomes sparser as the violation rate decreases. We therefore adaptively weight the rollout loss:

$$\kappa = \mathrm{sg}(\overline{\mathcal{L}_{EDM}})/(\mathrm{sg}(\overline{\mathcal{L}_{\mathrm{rollout}}}) + \epsilon), \epsilon = 10^{-5}, \tag{13}$$

where the overline denotes the batch mean, $\mathrm{sg}(\cdot)$ stops gradients through the scaling factors. The adaptive stop-gradient weight keeps the rollout loss on the same scale as the EDM loss, maintaining effective constraint supervision without introducing additional gradient pathways.

We summarize the full training procedure in Algorithm 1. At inference time, sampling follows the same reverse-time rollout, with the same noise discretization schedule used during training. The rollout starts from a Gaussian initialization at the highest noise level and iteratively denoises to obtain the final sample. The constraint-aware denoiser $D_\phi^\Omega$ is used for sampling, while the pretrained denoiser $D$ provides the denoised estimate for constraint gradient computation. Terminal loss evaluation and parameter updates are omitted.

### 4.4 Theoretical analysis

We formalize the training-time alignment by viewing sampling as a composition of differentiable denoising maps, showing that rollout backpropagation optimizes constraint violations under the same sampling distribution at inference.

**Theorem 4.3.** *(Training-time alignment) On the fixed schedule $\{t_i\}_{i=0}^N$, let each denoising step be a map $\mathbf{x}_{i-1} = f_\phi^{(i)}(\mathbf{x}_i; t_i, t_{i-1}, \epsilon_i), \epsilon_i \sim \mathcal{N}(0, \boldsymbol{I})$ and $f_\phi$ is differentiable in $\mathbf{x}_i$ and $\phi$. For an initial noise sample $\mathbf{x}_N$, the terminal sample is obtained by composing the update maps: $F_\phi = f_\phi^{(1)} \circ f_\phi^{(2)} \circ \cdots \circ f_\phi^{(N)}$ such that $\mathbf{x}_0^\phi = F_\phi(\mathbf{x}_N)$ is the terminal state. Then the gradient of unscaled rollout objective Equation* (12) *$\mathcal{L}_{\mathrm{rollout}}(\phi)$ with respect to $\phi$ is:*

$$\nabla_\phi \mathcal{L}_{\mathrm{rollout}}(\phi) = \mathbb{E}_{p(\mathbf{x}_N)}\left[\sum_{i=1}^N \nabla_{\mathbf{x}_0^\phi}\ell^\Omega(\mathbf{x}_0^\phi)\frac{\partial \mathbf{x}_0^\phi}{\partial \mathbf{x}_{i-1}^\phi}\frac{\partial f_\phi^{(i)}(\mathbf{x}_i)}{\partial \phi}\right]. \tag{14}$$

*This concludes that backpropagating $\ell^\Omega$ through the rollout computes the exact gradient of $\mathcal{L}_{\mathrm{rollout}}(\phi)$, directly minimizing constraint violations under the model's own sampling distribution.*

**Remark 4.4.** *(Full training objective) The full objective in Equation* (10) *augments the rollout loss with $\mathcal{L}_{\mathrm{EDM}}(\phi)$. Since $\mathcal{L}_{\mathrm{EDM}}(\phi)$ is evaluated on clean data samples independently of the rollout trajectory, its gradient is simply an additive term outside the rollout computation graph. Thus, Theorem 4.3 still applies to $\mathcal{L}(\phi)$, with $\nabla_\phi \mathcal{L}_{\mathrm{EDM}}(\phi)$ added independently.*

## 5 Experiments

We evaluate whether IMCD improves constraint enforcement while preserving generative fidelity under the same experimental conditions as prior work. To enable direct comparison, we adopt the

experimental setup of (Liang et al., 2026) and focus on isolating the effect of IMCD. Specifically, we assess whether the proposed design reduces infeasible generation through rollout-based training and improves stability across noise levels.

### 5.1 BOUNCING BALLS

We begin with a synthetic bouncing ball experiment, where multiple balls move inside a closed box with elastic ball-ball and ball-wall collisions. This setting enables controlled evaluation of both physical consistency and constraint satisfaction. Following (Liang et al., 2026), we generate simulated trajectories with randomized initial positions and velocities using a physics-based simulator (Gan et al., 2015). The dataset contains 100,000 scenarios, each with 10 balls over 100 timesteps. The goal is to model the trajectory distribution while enforcing two constraints: (i) boundary constraints, keeping all balls inside the box, (ii) overlap constraints, preventing overlap between balls.

We compare against unconstrained EDM (Karras et al., 2022) and three classes of constrained methods. The forward-state fine-tuning methods are MBM (Naderiparizi et al., 2025), MBM++ (Liang et al., 2026), and PIDM (Bastek et al., 2024), the last evaluated both trained from scratch and fine-tuned (FT). The inference-time methods operate on frozen EDM: MPGD (He et al., 2024), adapted with the MBM++ scaling schedule and without decoder projection; FK-Steering (Singhal et al., 2025) and D-Flow (Ben-Hamu et al., 2024). The rollout-based fine-tuning methods are Adjoint matching (AM) (Domingo-Enrich et al., 2024) and DPOK (Fan et al., 2023), both regularizing toward the pretrained model via KL. All methods use 50 reverse-time steps, and Section E.1 details each baseline.

We evaluate constraint satisfaction using overlap and boundary violation rates, and assess distributional fidelity using reweighted ELBO (r-ELBO). To capture physical motion consistency, we introduce three additional metrics. We measure maximum frame-to-frame displacement (F2F) to quantify temporal inconsistency and detect jittering artifacts, maximum contact distance (MCD) to capture worst-case proximity during ball-ball and ball-wall interactions, and maximum energy deviation (MED) to measure violations of kinetic energy conservation over time. We also compute these metrics on ground-truth trajectories for reference. Full definitions of these metrics are provided in the appendix Section E.2.

Table 1 reports results on the bouncing ball experiment. Baselines reveal a trade-off between feasibility, physical realism, and fidelity. Among inference-time methods, MPGD enforces constraints perfectly, but its high F2F and MED indicate abrupt frame-to-frame corrections and jittery motion, as visualized in Figure 2, whereas FK-Steering and D-Flow generate smoother trajectories but leave substantial constraint violations. MBM and MBM++ reduce violations but remain unstable or lose physical plausibility under few-step sampling. PIDM and PIDM (FT) preserve plausible dynamics but fail to enforce hard constraints, while DPOK and AM either degrade fidelity or only partially satisfy constraints. Our method avoids these trade-offs: it achieves near-zero boundary and overlap violations while keeping F2F and MED close to the EDM baseline, indicating smooth constraint-aware rollouts rather than abrupt corrections. Its competitive fidelity shows that retaining EDM denoising preserves the training data distribution, yielding the strongest overall balance across metrics. To better understand these gains, we further provide ablation studies on the effect of rollout-based training and the trainable components in Section E.3.

### 5.2 TRAFFIC SCENE TRAJECTORY PREDICTION

We next evaluate real-world traffic scene trajectory prediction on the Interaction dataset (Zhan et al., 2019), which contains vehicle trajectories from 11 traffic scenarios. Given a short history of observed states, the model predicts future positions and headings for all agents in the scene. This setting tests multi-agent driving behavior modeling under road geometry and vehicle interactions, including staying within drivable areas and avoiding collisions. Following prior work (Lioutas et al., 2023; Liang et al., 2026), we observe one second of motion and predict a three-second future horizon.

We reuse the bouncing ball baselines plus CriticSMC (Lioutas et al., 2023). We omit MBM, which MBM++ dominates, along with AM, FK-Steering, and D-Flow, whose slow convergence, particle populations, and per-sample optimization do not scale here.

We use DJINN (Niedoba et al., 2024), an EDM-based diffusion model for joint trajectory prediction, as the pretrained backbone, fine-tuning it with the corresponding objective for baselines that require

Table 1: Bouncing ball experiment results. We report constraint violation rates, distributional fidelity (r-ELBO), and physical consistency metrics (F2F, MCD, MED). Ground truth values are provided for reference.

| | Constraint (%) ↓ | | Fidelity ↑ | Physical plausibility ($\times 10^{-1}$) ↓ | | |
|---|---|---|---|---|---|---|
| Method | Boundary rate | Overlap rate | r-ELBO ($\times 10^{-2}$) | F2F | MCD | MED |
| EDM | $6.01 \pm 0.02$ | $39.82 \pm 0.21$ | $-21.7 \pm 0.1$ | $3.1 \pm 0.0$ | $1.0 \pm 0.0$ | $0.4 \pm 0.0$ |
| MPGD | $0.00 \pm 0.00$ | $0.00 \pm 0.00$ | $-27.0 \pm 0.1$ | $8.6 \pm 0.4$ | $0.8 \pm 0.0$ | $29.7 \pm 5.4$ |
| FK-Steering | $6.13 \pm 0.03$ | $16.72 \pm 0.05$ | $-21.7 \pm 0.1$ | $3.1 \pm 0.0$ | $0.8 \pm 0.0$ | $0.4 \pm 0.0$ |
| D-Flow | $5.20 \pm 0.11$ | $35.92 \pm 0.30$ | $-21.7 \pm 0.1$ | $3.1 \pm 0.0$ | $1.1 \pm 0.0$ | $0.4 \pm 0.0$ |
| MBM | $0.36 \pm 0.00$ | $0.16 \pm 0.00$ | $-22.2 \pm 0.1$ | $59.7 \pm 0.4$ | $1.7 \pm 0.0$ | $510.9 \pm 11.5$ |
| MBM++ | $0.00 \pm 0.00$ | $0.00 \pm 0.00$ | $-22.8 \pm 0.1$ | $10.3 \pm 0.6$ | $0.8 \pm 0.0$ | $41.1 \pm 5.9$ |
| PIDM | $11.07 \pm 0.15$ | $40.91 \pm 0.19$ | $-26.9 \pm 0.1$ | $2.9 \pm 0.0$ | $1.1 \pm 0.0$ | $0.3 \pm 0.0$ |
| PIDM (FT) | $6.40 \pm 0.04$ | $29.15 \pm 0.10$ | $-21.6 \pm 0.1$ | $3.0 \pm 0.0$ | $0.8 \pm 0.0$ | $0.3 \pm 0.0$ |
| AM | $0.49 \pm 0.01$ | $4.64 \pm 0.06$ | $-39.4 \pm 0.2$ | $3.4 \pm 0.0$ | $3.8 \pm 0.0$ | $0.5 \pm 0.0$ |
| DPOK | $2.04 \pm 0.05$ | $1.69 \pm 0.03$ | $-53.5 \pm 0.1$ | $2.8 \pm 0.0$ | $4.3 \pm 0.0$ | $0.3 \pm 0.0$ |
| IMCD (**Ours**) | $0.01 \pm 0.00$ | $0.01 \pm 0.00$ | $-22.9 \pm 0.0$ | $3.4 \pm 0.1$ | $1.2 \pm 0.0$ | $0.6 \pm 0.0$ |
| Ground truth | $0.00 \pm 0.00$ | $0.00 \pm 0.00$ | – | $2.8 \pm 0.0$ | $0.4 \pm 0.0$ | $0.3 \pm 0.0$ |

Table 2: Traffic scene trajectory prediction results on the INTERACTION DR_DEU_Merging_MT scenario. $ADE_6$, $FDE_6$, $SADE_6$, and $SFDE_6$ denote the minimum-over-6-samples trajectory-match metrics (i.e., $minADE_6$, $minFDE_6$, $minSADE_6$, $minSFDE_6$). Ground truth values are provided for reference.

| | Constraint (%) ↓ | | Min traj. match ↓ | | | | Phys. plaus. ↓ | | Diversity ↑ |
|---|---|---|---|---|---|---|---|---|---|
| Method | Offroad | Collision | $ADE_6$ | $FDE_6$ | $SADE_6$ | $SFDE_6$ | $SF2F_6$ | $SMLV_6$ | $MFD_6$ |
| EDM | 9.03 | 0.42 | 0.20 | 0.50 | 0.27 | 0.77 | 1.11 | 0.41 | 2.75 |
| MPGD | 3.78 | 0.29 | 0.19 | 0.46 | 0.23 | 0.67 | 1.04 | 0.34 | 1.92 |
| MBM++ | 3.73 | 0.29 | 0.17 | 0.45 | 0.22 | 0.66 | 1.01 | 0.31 | 1.88 |
| PIDM (FT) | 8.52 | 0.34 | 0.18 | 0.50 | 0.24 | 0.76 | 0.96 | 0.26 | 2.67 |
| DPOK | 5.45 | 0.03 | 3.45 | 9.94 | 3.34 | 9.73 | 0.88 | 0.40 | 1.90 |
| CriticSMC | – | 1.03 | 0.35 | – | – | – | – | – | 2.37 |
| IMCD (**Ours**) | 0.30 | 0.00 | 0.18 | 0.45 | 0.23 | 0.66 | 1.07 | 0.43 | 1.91 |
| Ground truth | 0.00 | 0.00 | – | – | – | – | 0.83 | 0.02 | – |

training and for our method. We report our main variant in the main text and defer additional rollout/LoRA variants to Section F.2.

For evaluation, we generate six trajectory samples per scene and report metrics covering constraint satisfaction, trajectory match, physical plausibility and diversity. Constraint satisfaction is measured by offroad and collision rates. Trajectory match is evaluated using ego $minADE_6$/$minFDE_6$, where scene metrics $minSADE_6$/$minSFDE_6$ are computed over all vehicles. Physical plausibility is measured by maximum scene frame-to-frame displacement (SF2F) and maximum scene lateral velocity (SMLV), which capture abrupt motion and unrealistic lateral shift. Diversity is measured by maximum ego final distance ($MFD_6$) across six samples.

Table 2 reports results on the INTERACTION DR_DEU_Merging_MT scenario. Existing methods remain limited in different ways: EDM and PIDM (FT) better preserve diversity and physical plausibility but retain substantial violations, while MPGD and MBM++ reduce violations but still incur non-negligible offroad errors. In contrast, our IMCD achieves the strongest feasibility, substantially reducing offroad violations and eliminating collisions while maintaining good trajectory match. Its physical plausibility metrics remain close to EDM, indicating that constraint satisfaction is achieved without severe motion distortion. Overall, IMCD gives the best balance between feasibility and quality under few-step generation. Additional experiments and visualizations on different locations are reported in Section F.5.

# 6 Related Work

## 6.1 Diffusion models for constrained generation

Constrained diffusion has been approached by modifying either the forward or reverse process. Liu & Wu (2023) employ Doob's h-transform (Rogers & Williams, 2000) to construct a drift confining the process to the constraint domain at training time, while reflected diffusion (Lou & Ermon, 2023) and Fishman et al. (2023) alter the forward noising process through reflected Brownian motion or log-barrier metrics. Projected diffusion (Christopher et al., 2024), ECI (Cheng et al., 2025) and PCFM (Utkarsh et al., 2026) instead enforce constraints at inference time by projecting noisy samples onto the constraint set at each denoising step, with extensions to multi-robot motion planning (Liang et al., 2025). Beyond distorting the denoising trajectory, this requires a tractable projection onto the feasible set, which our constraints do not admit in closed form, making these methods less applicable here. IMCD instead incorporates constraint correction into training through rollout, allowing the denoiser to adapt to the corrected sampling process rather than treating correction as a post hoc perturbation to a fixed pretrained model.

## 6.2 Reward-driven fine-tuning and controlled diffusion

Fine-tuning diffusion models with reward or energy objectives is closely related to our training formulation. We defer a detailed comparison with AM and DPOK to Section C. A related line of work views diffusion sampling as stochastic optimal control, where methods such as AM, Adjoint sampling (AS) (Havens et al., 2025) and Adjoint Schrodinger bridge Sampler (ASBS) (Liu et al., 2025) learn reverse-time controls that drive samples toward reward- or energy-defined targets. ASBS further extends AS to arbitrary source distributions. However, these methods are designed to optimize reward or energy objectives directly, rather than leverage a pretrained model as the base distribution. In contrast, our goal is constrained generation, which fine-tunes a pretrained denoiser through its sampling rollout to improve feasibility while preserving the learned data distribution.

## 6.3 Diffusion models for autonomous driving

Several recent works have explored diffusion models for autonomous driving planning. DJINN (Niedoba et al., 2024) jointly diffuses trajectories of all agents for realistic scenario generation, but is primarily designed for simulation rather than planning, and does not focus on constraint satisfaction. Diffusion-ES (Yang et al., 2024a) combines gradient-free evolutionary search with a diffusion trajectory model to optimize non-differentiable test-time reward objectives, but operates on the ego vehicle only without modeling interactions with other agents. Diffusion Planner (Zheng et al., 2025) jointly models ego planning and agent prediction with a unified diffusion architecture, and applies classifier guidance at inference time to achieve constraint satisfaction, but may encounter the sampling-time correction limitation discussed in Section 3.2.

# 7 Discussion and future work

Our method IMCD provides a rollout-based fine-tuning framework for constrained diffusion generation that addresses training-time misalignment. By minimizing constraint violations on terminal rollout samples, the model learns corrections under its own sampling dynamics rather than relying on post hoc guidance. Both theory and experiments show that our method improves constraint satisfaction while preserving distributional fidelity across bouncing ball simulation and autonomous driving tasks. While effective, the current inference procedure still involves constraint-gradient computation during the reverse rollout, adding modest computational overhead. A promising direction is to improve efficiency through distillation, where a standalone denoiser learns to approximate the constraint-aware updates directly. Another important direction is to extend rollout-based fine-tuning to higher-dimensional visual domains and broader application settings where feasibility is specified by task-specific violation signals.

### AI USE STATEMENT

In this work, we did not use generative AI tools for any task on the required-disclosure list. We did not use generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. Additionally, we used generative AI tools to edit the manuscript and correct grammar mistakes. We have reviewed all AI-assisted work. Each suggested edit was meant to improve readability, so that no technical statement is changed. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

Our method IMCD reduces constraint violations but does not eliminate them across traffic scenarios. Therefore it should be treated as a way to improve the feasibility of learned trajectory distributions, not as a safety mechanism. A deployed system would still require independent verification and hard constraint enforcement downstream of generation. We identify no other concerns relative to the Code of Ethics.

### REPRODUCIBILITY STATEMENT

We describe the experimental setup and results in Section 5 and provide full details in Appendices E and F, including baseline configurations, constraint functions and metric definitions, noise schedules, optimizers, learning rates, and training budgets. The complete training procedure is given in Algorithm 1 (Appendix A), and the scaling network architecture, its parameterization, and the multi-constraint gradient projection are described in Appendix D and Sections E.5 and F.3. Training- and inference-time compute costs are reported in Sections E.6 and E.7. For theoretical results, all assumptions are stated explicitly, and complete proofs are provided in Appendices B and C.

## REFERENCES

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Jan-Hendrik Bastek, WaiChing Sun, and Dennis M Kochmann. Physics-informed diffusion models. *arXiv preprint arXiv:2403.14404*, 2024.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

Giulio Biroli, Tony Bonnaire, Valentin De Bortoli, and Marc Mézard. Dynamical regimes of diffusion models. *Nature Communications*, 15(1):9957, 2024.

Benjamin Boys, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and O Deniz Akyildiz. Tweedie moment projected diffusions for inverse problems. *arXiv preprint arXiv:2310.06721*, 2023.

Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1916–1923. IEEE, 2023.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Chaoran Cheng, Boran Han, Danielle Maddix, Abdul Fatir Ansari, Andrew Stuart, Michael W Mahoney, and Bernie Wang. Gradient-free generation for hard-constrained systems. In *International Conference on Learning Representations*, volume 2025, pp. 100510–100539, 2025.

Jacob K Christopher, Stephen Baek, and Ferdinando Fioretto. Projected generative diffusion models for constraint satisfaction. *arXiv preprint arXiv:2402.03559*, 2024.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. Cfg++: Manifold-constrained classifier free guidance for diffusion models. *arXiv preprint arXiv:2406.08070*, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*, 2024.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

Nic Fishman, Leo Klarner, Valentin De Bortoli, Emile Mathieu, and Michael John Hutchinson. Diffusion models for constrained domains. *Transactions on Machine Learning Research*, 2023.

Zhe Gan, Chunyuan Li, Ricardo Henao, David E Carlson, and Lawrence Carin. Deep temporal sigmoid belief networks for sequence modeling. *Advances in Neural Information Processing Systems*, 28, 2015.

William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *Advances in neural information processing systems*, 35: 27953–27965, 2022.

Aaron Havens, Benjamin Kurt Miller, Bing Yan, Carles Domingo-Enrich, Anuroop Sriram, Brandon Wood, Daniel Levine, Bin Hu, Brandon Amos, Brian Karrer, et al. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. *arXiv preprint arXiv:2504.11713*, 2025.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Mingxiao Li, Tingyu Qu, Ruicong Yao, Wei Sun, and Marie-Francine Moens. Alleviating exposure bias in diffusion models through sampling with shifted time steps. *arXiv preprint arXiv:2305.15583*, 2023.

Jinhao Liang, Jacob K Christopher, Sven Koenig, and Ferdinando Fioretto. Simultaneous multi-robot motion planning with projected diffusion models. *arXiv preprint arXiv:2502.03607*, 2025.

Xiaoxuan Liang, Saeid Naderiparizi, Yunpeng Liu, Berend Zwartsenberg, and Frank Wood. Improved constrained generation by bridging pretrained generative models. *arXiv preprint arXiv:2603.06742*, 2026.

Vasileios Lioutas, Jonathan Wilder Lavington, Justice Sefas, Matthew Niedoba, Yunpeng Liu, Berend Zwartsenberg, Setareh Dabiri, Frank Wood, and Adam Scibior. Critic sequential monte carlo. In *The Eleventh International Conference on Learning Representations*, 2023.

Guan-Horng Liu, Jaemoo Choi, Yongxin Chen, Benjamin Kurt Miller, and Ricky TQ Chen. Adjoint schr\" odinger bridge sampler. *arXiv preprint arXiv:2506.22565*, 2025.

Xingchao Liu and Lemeng Wu. Learning diffusion bridges on constrained domains. In *international conference on learning representations (ICLR)*, 2023.

Aaron Lou and Stefano Ermon. Reflected diffusion models. In *International Conference on Machine Learning*, pp. 22675–22701. PMLR, 2023.

Saeid Naderiparizi, Xiaoxuan Liang, Setareh Cohan, Berend Zwartsenberg, and Frank Wood. Don't be so negative! score-based generative modeling with oracle-assisted guidance. In *Forty-first International Conference on Machine Learning*, 2024.

Saeid Naderiparizi, Xiaoxuan Liang, Berend Zwartsenberg, and Frank Wood. Constrained generative modeling with manually bridged diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19607–19615, 2025.

Matthew Niedoba, Jonathan Lavington, Yunpeng Liu, Vasileios Lioutas, Justice Sefas, Xiaoxuan Liang, Dylan Green, Setareh Dabiri, Berend Zwartsenberg, Adam Scibior, et al. A diffusion-model of joint interactive navigation. *Advances in Neural Information Processing Systems*, 36, 2024.

Mang Ning, Enver Sangineto, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Input perturbation reduces exposure bias in diffusion models. *arXiv preprint arXiv:2301.11706*, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Lyle Regenwetter, Giorgio Giannone, Akash Srivastava, Dan Gutfreund, and Faez Ahmed. Constraining generative models for engineering design with negative data. *Transactions on Machine Learning Research*, 2024.

Leonard CG Rogers and David Williams. *Diffusions, markov processes, and martingales: Volume 1, foundations*. Cambridge university press, 2000.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

Yifei Shen, Xinyang Jiang, Yifan Yang, Yezhen Wang, Dongqi Han, and Dongsheng Li. Understanding and improving training-free loss-based diffusion guidance. *Advances in Neural Information Processing Systems*, 37:108974–109002, 2024.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

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

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Utkarsh Utkarsh, Pengfei Cai, Alan Edelman, Rafael Gomez-Bombarelli, and Christopher Rackauckas. Physics-constrained flow matching: Sampling generative models with hard constraints. *Advances in Neural Information Processing Systems*, 38:160217–160252, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-es: Gradient-free planning with diffusion for autonomous driving and zero-shot instruction following. *arXiv preprint arXiv:2402.06559*, 2024a.

Lingxiao Yang, Shutong Ding, Yifan Cai, Jingyi Yu, Jingya Wang, and Ye Shi. Guidance with spherical gaussian constraint for conditional diffusion. *arXiv preprint arXiv:2402.03201*, 2024b.

Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Zou, and Stefano Ermon. Tfg: Unified training-free guidance for diffusion models. *Advances in Neural Information Processing Systems*, 37:22370–22417, 2024.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23174–23184, 2023.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *arXiv preprint arXiv:2208.15001*, 2022.

Yinan Zheng, Ruiming Liang, Kexin Zheng, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, et al. Diffusion-based planning for autonomous driving with flexible guidance. *arXiv preprint arXiv:2501.15564*, 2025.

# A ALGORITHM

**Algorithm 1** Rollout-Based Constrained Fine-Tuning

**Require:** Pretrained denoiser $D$, fine-tuned denoiser $D_\phi$ initialized from $D$, scaling network $\gamma_\phi$, constraint loss $\ell^\Omega$, noise schedule $\{t_i\}_{i=0}^N$, training data $\mathcal{D}$, division stabilizer $\epsilon = 10^{-5}$.

**while** not converged **do**
  **// Rollout**
  Sample $\mathbf{x}_N \sim \mathcal{N}(0, t_N^2 \boldsymbol{I})$
  **for** $i = N, \dots, 1$ **do**
    Compute constraint-aware fine-tuned denoiser $D_\phi^\Omega(\tilde{\mathbf{x}}_i; t_i)$ with Equation (9).
    Update the rollout state to $\tilde{\mathbf{x}}_{i-1}$.
  **end for**
  **// Losses**
  $\mathcal{L}_{\text{rollout}} \leftarrow \ell^\Omega(\tilde{\mathbf{x}}_0)$ ▷ Terminal state only
  Sample $\mathbf{x}_0 \sim \mathcal{D}$, $\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0)$
  $\mathcal{L}_{\text{EDM}} \leftarrow \lambda(t) \| D_\phi^\Omega(\mathbf{x}_t; t) - \mathbf{x}_0 \|^2$
  **// Dynamic Scaling and Update**
  $\kappa \leftarrow \frac{\text{sg}(\mathcal{L}_{\text{EDM}})}{\text{sg}(\mathcal{L}_{\text{rollout}}) + \epsilon}$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{EDM}} + \kappa \mathcal{L}_{\text{rollout}}$
  Update $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}$ ▷ Gradients flow only through $\phi$
**end while**

**Gradient embeddings** Following MBM++, the constraint gradient is embedded with a lightweight MLP and added once to $\mathbf{x}_t$ as input to $D_\phi(\mathbf{x}_t; t)$. Since this modifies the model input distribution seen by the pretrained backbone, we introduce LoRA adapters (Hu et al., 2022) in the attention layers of $D_\phi$ to provide the necessary adaptation capacity.

Table 3: A summary table of scaling and guidance functions for various methods.

| | MBM (Naderiparizi et al., 2025) | MBM++ (Liang et al., 2026) | IMCD (ours) |
|---|---|---|---|
| $\gamma(\mathbf{x}_t; t)$ | $c/t^2$ | $c/t^2$ | $\alpha_\phi(\mathbf{x}_t; t) \cdot t^{\beta_\phi(\mathbf{x}_t; t)}$ |
| $G^\Omega(\mathbf{x}_t; t)$ | $\partial_1 \ell^\Omega(\mathbf{x}_t; t)$ | $\partial_1 \ell^\Omega(D(\mathbf{x}_t; t); t)$ | $\partial_1 \ell^\Omega(D(\mathbf{x}_t; t); t)$ |

# B DERIVATION OF PROOFS

## B.1 PROOF OF PROPOSITION 4.2

*Proof.* Given the formulation of $p^\Omega(\mathbf{x}_t; t)$, take gradient $\nabla_{\mathbf{x}_t} \log p^\Omega(\mathbf{x}_t; t)$ and treat $\gamma_\phi(\mathbf{x}_t; t)$ as constant with respect to $\mathbf{x}_t$. This yields the score decomposition in Definition 4.1. For the convergence of denoising terminal state to the constrained region, since $\beta_\phi < -1$, $t^{\beta_\phi} \to \infty$ as $t \to 0$. Given $\alpha_\phi > 0$, we have $\gamma_\phi(\mathbf{x}_t; t) \to \infty$ as $t \to 0$. Then

$$\lim_{t \to 0} \exp(-\gamma_\phi(\mathbf{x}_t; t) \ell^\Omega(D(\mathbf{x}_t; t))) = \begin{cases} 1 & \text{if } \ell^\Omega(D(\mathbf{x}_t; t)) = 0 \\ 0 & \text{if } \ell^\Omega(D(\mathbf{x}_t; t)) > 0 \end{cases} \tag{15}$$

Therefore, with $\lim_{t \to 0} D(\mathbf{x}_t; t) = \mathbf{x}_t$,

$$\lim_{t \to 0} p^\Omega(\mathbf{x}_t; t) \propto \lim_{t \to 0} p(\mathbf{x}_t; t) \mathbf{1}_\Omega(\mathbf{x}_t), \tag{16}$$

meaning that as $t \to 0$, the distribution concentrates on the constrained region $\Omega$, ensuring that the terminal denoised sample $\mathbf{x}_0 \sim p(\mathbf{x}_0, 0)$ lies within $\Omega$. □

### B.2 Proof of Theorem 4.3

*Proof.* Since each $\epsilon_i$ is independent of $\phi$, the trajectory $\{\mathbf{x}_i\}_{i=0}^N$ is a deterministic differentiable function of $\phi$. Because each $f_\phi^{(i)}$ is differentiable, then the composed map $F_\phi$ is differentiable. By chain rule,

$$\nabla_\phi \ell^\Omega(F_\phi(\mathbf{x}_N)) = \nabla_{\mathbf{x}_0^\phi} \ell^\Omega(\mathbf{x}_0^\phi) \frac{\partial F_\phi(\mathbf{x}_N)}{\partial \phi}. \tag{17}$$

Then the gradient of $\mathcal{L}_{\text{rollout}}(\phi)$ with respect to $\phi$ is

$$\nabla_\phi \mathcal{L}_{\text{rollout}}(\phi) = \nabla_\phi \mathbb{E}_{p(\mathbf{x}_N)} \left[\ell^\Omega(F_\phi(\mathbf{x}_N))\right] = \mathbb{E}_{p(\mathbf{x}_N)} \left[\nabla_\phi \ell^\Omega(F_\phi(\mathbf{x}_N))\right] \tag{18}$$

$$= \mathbb{E}_{p(\mathbf{x}_N)} \left[\nabla_{\mathbf{x}_0^\phi} \ell^\Omega(\mathbf{x}_0^\phi) \frac{\partial F_\phi(\mathbf{x}_N)}{\partial \phi}\right] \tag{19}$$

by interchanging differentiation with expectation.

Now we expand $\frac{\partial F_\phi(\mathbf{x}_N)}{\partial \phi}$ by applying the chain rule through the individual step $f_\phi^{(i)}$:

$$\frac{\partial F_\phi(\mathbf{x}_N)}{\partial \phi} = \sum_{i=1}^N \frac{\partial \mathbf{x}_0^\phi}{\partial \mathbf{x}_i^\phi} \cdot \frac{\partial \mathbf{x}_i^\phi}{\partial \phi}, \tag{20}$$

where $\frac{\partial \mathbf{x}_0^\phi}{\partial \mathbf{x}_i^\phi}$ is the Jacobian of the terminal state $\mathbf{x}_0^\phi$ with respect to the intermediate state $\mathbf{x}_i^\phi$. The second term $\frac{\partial \mathbf{x}_i^\phi}{\partial_\phi}$, where $\mathbf{x}_i^\phi = f_\phi^{(i+1)}(\mathbf{x}_{i+1})$, can be further decomposed to:

$$\frac{\partial \mathbf{x}_i^\phi}{\partial \phi} = \frac{\partial f_\phi^{(i+1)}(\mathbf{x}_{i+1})}{\partial \mathbf{x}_{i+1}^\phi} \cdot \frac{\partial \mathbf{x}_{i+1}^\phi}{\partial \phi} + \frac{\partial f_\phi^{(i+1)}(\mathbf{x}_{i+1})}{\partial \phi}. \tag{21}$$

Given this recursive pattern with base case $\frac{\partial \mathbf{x}_N}{\partial \phi} = 0$ as $\mathbf{x}_N$ is independent of $\phi$, we have

$$\frac{\partial \mathbf{x}_{N-1}^\phi}{\partial \phi} = \frac{\partial f_\phi^{(N)}(\mathbf{x}_N)}{\partial \mathbf{x}_N^\phi} \cdot \frac{\partial \mathbf{x}_N^\phi}{\partial \phi} + \frac{\partial f_\phi^{(N)}(\mathbf{x}_N)}{\partial \phi} = \frac{\partial f_\phi^{(N)}(\mathbf{x}_N)}{\partial \phi}, \tag{22}$$

$$\frac{\partial \mathbf{x}_{N-2}^\phi}{\partial \phi} = \frac{\partial f_\phi^{(N-1)}(\mathbf{x}_{N-1})}{\partial \mathbf{x}_{N-1}^\phi} \cdot \frac{\partial \mathbf{x}_{N-1}^\phi}{\partial \phi} + \frac{\partial f_\phi^{(N-1)}(\mathbf{x}_{N-1})}{\partial \phi} \tag{23}$$

$$= \frac{\partial f_\phi^{(N-1)}(\mathbf{x}_{N-1})}{\partial \mathbf{x}_{N-1}^\phi} \cdot \frac{\partial f_\phi^{(N)}(\mathbf{x}_N)}{\partial \phi} + \frac{\partial f_\phi^{(N-1)}(\mathbf{x}_{N-1})}{\partial \phi} \tag{24}$$

$$= \frac{\partial \mathbf{x}_{N-2}^\phi}{\partial \mathbf{x}_{N-1}^\phi} \cdot \frac{\partial f_\phi^{(N)}(\mathbf{x}_N)}{\partial \phi} + \frac{\partial f_\phi^{(N-1)}(\mathbf{x}_{N-1})}{\partial \phi}. \tag{25}$$

This gives a general form for $i \in \{0, 1, \cdots, N-1\}$:

$$\frac{\partial \mathbf{x}_i^\phi}{\partial \phi} = \sum_{j=i+1}^N \frac{\partial \mathbf{x}_i^\phi}{\partial \mathbf{x}_{j-1}^\phi} \cdot \frac{\partial f_\phi^{(j)}(\mathbf{x}_j)}{\partial \phi}. \tag{26}$$

Then

$$\frac{\partial F_\phi(\mathbf{x}_N)}{\partial \phi} = \frac{\partial \mathbf{x}_0^\phi}{\partial \phi} = \sum_{i=1}^N \frac{\partial \mathbf{x}_0^\phi}{\partial \mathbf{x}_{i-1}^\phi} \cdot \frac{\partial f_\phi^{(i)}(\mathbf{x}_i)}{\partial \phi}. \tag{27}$$

Therefore,

$$\nabla_\phi \mathcal{L}_{\text{rollout}}(\phi) = \mathbb{E}_{p(\mathbf{x}_N)} \left[\sum_{i=1}^N \nabla_{\mathbf{x}_0^\phi} \ell^\Omega(\mathbf{x}_0^\phi) \frac{\partial \mathbf{x}_0^\phi}{\partial \mathbf{x}_{i-1}^\phi} \frac{\partial f_\phi^{(i)}(\mathbf{x}_i)}{\partial \phi}\right]. \tag{28}$$

□

**Corollary B.1.** *(Unbiased gradient estimator) Let $\mathbf{x}_N$ be a single initial noise sample and let $\{\mathbf{x}_i^\phi\}_{i=0}^{N-1}$ be the trajectory by rolling out $\{f_\phi^{(i)}\}_{i=1}^N$. Then the single sample estimator*

$$h(\phi) = \sum_{i=1}^{N} \nabla_{\mathbf{x}_0^\phi} \ell^\Omega(\mathbf{x}_0^\phi) \frac{\partial \mathbf{x}_0^\phi}{\partial \mathbf{x}_{i-1}^\phi} \frac{\partial f_\phi^{(i)}(\mathbf{x}_i)}{\partial \phi} \tag{29}$$

*is an unbiased estimator of $\nabla_\phi \mathcal{L}_{\text{rollout}}(\phi)$.*

*Proof.* This is an immediate conclusion from Theorem 4.3. Since $h(\phi)$ is a fixed realization, this result is obtained by taking expectation over $p(\mathbf{x}_N)$ and interchanging expectation and differentiation, which is,

$$\nabla_\phi \mathcal{L}_{\text{rollout}}(\phi) = \mathbb{E}_{p(\mathbf{x}_N)} \left[ h(\phi) \right]. \tag{30}$$

□

# C Rollout-based fine-tuning and stochastic optimal control

Our rollout-based training admits a natural stochastic optimal control (SOC) interpretation: the reverse denoising process induced by the pretrained denoiser $D(\mathbf{x}_t; t)$ and constraint-aware fine-tuned denoiser $D_\phi^\Omega$ correspond to the base and controlled stochastic processes respectively, and the rollout constraint loss acts as a terminal cost. This connects our approach to Adjoint Matching (AM) (Domingo-Enrich et al., 2024), which provides a formal SOC grounding for reward fine-tuning of generative models via rollout-based optimization. It is also related to DPOK (Fan et al., 2023), which frames the denoising chain as a Markov decision process (MDP) and applies policy gradient with KL regularization. Both AM and DPOK approaches regularize the fine-tuned model by minimizing its KL divergence from the pretrained model, treating the pretrained model as the reference. AM additionally incurs significant computational overhead beyond rollout simulation, as each gradient update requires integrating the lean adjoint ODE backward through the trajectory. However, in our setting, the pretrained denoiser can produce samples that violate the constraints, so enforcing proximity to it can perpetuate these errors. Instead, we rely on initialization from the pretrained model, anchor to the data distribution directly via the denoising objective, and combine it with rollout-time constraint losses to drive the sampler toward the feasible manifold, preserving consistency with the data rather than with the pretrained model's potentially infeasible outputs. The next subsection shows this formally.

## C.1 EDM denoising loss as an alternative to reference KL

Let $p_\theta$ be the transition kernel induced by pretrained diffusion model $D_\theta$, and $p_\phi$ be the transition kernel induced by its fine-tuned model $D_\phi$. For fine-tuning, we compare the reference method DPOK (Fan et al., 2023) regularizer with its upper bound:

$$\mathcal{L}_{\text{KL}}(\phi) = \mathbb{E}_{p(\mathbf{z})} \left[ D_{\text{KL}}(p_\phi(\mathbf{x}_0 \mid \mathbf{z}) \parallel p_\theta(\mathbf{x}_0 \mid \mathbf{z})) \right] \tag{31}$$

$$\leq \mathbb{E}_{p(\mathbf{z})} \left[ \sum_{t=1}^{T} \mathbb{E}_{p_\phi(\mathbf{x}_t \mid \mathbf{z})} \left[ D_{\text{KL}} \left( p_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{z}) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{z}) \right) \right] \right], \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{32}$$

with our proposed EDM denoising regularizer

$$\mathcal{L}_{\text{EDM}}(\phi) = \mathbb{E}_{\mathbf{x}_0, t, \mathbf{z}} \left[ \lambda(t) \| D_\phi(\mathbf{x}_t; t) - \mathbf{x}_0 \|^2 \right], \quad \mathbf{x}_t = \mathbf{x}_0 + t\mathbf{z}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{33}$$

EDM is commonly sampled using a stochastic Heun sampler (Karras et al., 2022), a finite stepwise KL requires stochastic Gaussian transition. Therefore, we use stochastic reverse-SDE transitions (Song et al., 2021) with the same covariance for both models:

$$p_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mu_\phi(\mathbf{x}_t; t), \Sigma_t), \quad p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mu_\theta(\mathbf{x}_t; t), \Sigma_t). \tag{34}$$

Then it gives

$$\mathcal{L}_{\text{KL}}(\phi) \leq \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t \sim p_\phi} \left[ \frac{1}{2} \left( \mu_\phi(\mathbf{x}_t; t) - \mu_\theta(\mathbf{x}_t; t) \right)^\top \Sigma_t^{-1} \left( \mu_\phi(\mathbf{x}_t; t) - \mu_\theta(\mathbf{x}_t; t) \right) \right]. \tag{35}$$

For a fixed $(\mathbf{x}_t; t)$, the two regularizers have different minimizers:

$$D^*_{\mathrm{KL}}(\mathbf{x}_t; t) = \arg\min_D \|D - D_\theta(\mathbf{x}_t; t)\|^2 = D_\theta(\mathbf{x}_t; t), \tag{36}$$

$$D^*_{\mathrm{EDM}}(\mathbf{x}_t; t) = \arg\min_D \|D - \mathbf{x}_0\|^2 = \mathbb{E}\left[\mathbf{x}_0 \mid \mathbf{x}_t\right]. \tag{37}$$

Thus the two targets coincide only when the pretrained denoiser trains well:

$$D_\theta(\mathbf{x}_t; t) = \mathbb{E}\left[\mathbf{x}_0 \mid \mathbf{x}_t\right]. \tag{38}$$

**Proposition C.1** (EDM regularization vs. reference KL)**.** *We assume that $\ell^\Omega$ is L-Lipschitz, and every clean training sample $\mathbf{x}_0$ lies in the feasible set, i.e., $\ell^\Omega(\mathbf{x}_0) = 0$. Then the unweighted EDM error $\ell_{\mathrm{EDM}}(\phi) = \mathbb{E}\left[\|D_\phi(\mathbf{x}_t; t) - \mathbf{x}_0\|^2\right]$ controls expected violation as*

$$\mathbb{E}\left[\ell^\Omega(D_\phi(\mathbf{x}_t; t))\right] \le L\sqrt{\ell_{\mathrm{EDM}}(\phi)}, \quad \mathbf{x}_0 \sim p_0, \mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0). \tag{39}$$

*Moreover, at an Euler-Maruyama step $k$, if $\ell^\Omega(D_\theta(\mathbf{x}_k; t_k)) > 0$, any zero-violation prediction must satisfy $\|D_\phi(\mathbf{x}_k; t_k) - D_\theta(\mathbf{x}_k; t_k)\| \ge \ell^\Omega(D_\theta(\mathbf{x}_k; t_k))/L$. For the fixed EDM reverse-SDE sampler, this correction incurs*

$$D_{\mathrm{KL},k} \ge \frac{\Delta_k \left(\ell^\Omega(D_\theta(\mathbf{x}_k; t_k))\right)^2}{t_k^3 L^2}, \quad \Delta_k = t_k - t_{k-1}. \tag{40}$$

*Proof.* Given $\ell^\Omega$ is L-Lipschitz, we have

$$|\ell^\Omega(\mathbf{x}) - \ell^\Omega(\mathbf{y})| \le L\|\mathbf{x} - \mathbf{y}\|. \tag{41}$$

Substituting $\mathbf{x} = D_\phi(\mathbf{x}_t; t)$ and $\mathbf{y} = \mathbf{x}_0$ into the Lipschitz condition gives:

$$|\ell^\Omega(D_\phi(\mathbf{x}_t; t)) - \ell^\Omega(\mathbf{x}_0)| = \ell^\Omega(D_\phi(\mathbf{x}_t; t)) \le L\|D_\phi(\mathbf{x}_t; t) - \mathbf{x}_0\|, \tag{42}$$

since $\ell^\Omega(\mathbf{x}_0) = 0, \ell^\Omega(D_\phi(\mathbf{x}_t; t)) \ge 0$. Then we take expectation and then apply Cauchy-Schwarz,

$$0 \le \mathbb{E}\left[\ell^\Omega(D_\phi(\mathbf{x}_t; t))\right] \tag{43}$$

$$\le L\mathbb{E}\left[\|D_\phi(\mathbf{x}_t; t) - \mathbf{x}_0\|\right] \tag{44}$$

$$\le L\sqrt{\underbrace{\mathbb{E}\left[\|D_\phi(\mathbf{x}_t; t) - \mathbf{x}_0\|^2\right]}_{\ell_{\mathrm{EDM}}(\phi)}}. \tag{45}$$

Now we have

$$0 \le \mathbb{E}\left[\ell^\Omega(D_\phi(\mathbf{x}_t; t))\right] \le L\sqrt{\ell_{\mathrm{EDM}}(\phi)}. \tag{46}$$

Thus a small EDM loss forces the expected constraint violation to be small.

Given a particular $(x_t; t)$, suppose

$$\ell^\Omega(D_\theta(\mathbf{x}_t; t)) > 0, \tag{47}$$

then the Lipschitz continuity implies

$$|\ell^\Omega(D_\phi(\mathbf{x}_t; t)) - \ell^\Omega(D_\theta(\mathbf{x}_t; t))| \le L\|D_\phi(\mathbf{x}_t; t) - D_\theta(\mathbf{x}_t; t)\| \tag{48}$$

$$\Rightarrow \ell^\Omega(D_\phi(\mathbf{x}_t; t)) \ge [\ell^\Omega(D_\theta(\mathbf{x}_t; t)) - L\|D_\phi(\mathbf{x}_t; t) - D_\theta(\mathbf{x}_t; t)\|]_+ \tag{49}$$

If $\ell^\Omega(D_\phi(\mathbf{x}_t; t)) = 0$, then

$$\ell^\Omega(D_\theta(\mathbf{x}_t; t)) \le L\|D_\phi(\mathbf{x}_t; t) - D_\theta(\mathbf{x}_t; t)\| \Rightarrow \|D_\phi(\mathbf{x}_t; t) - D_\theta(\mathbf{x}_t; t)\| \ge \frac{\ell^\Omega(D_\theta(\mathbf{x}_t; t))}{L}. \tag{50}$$

Let $d_t = \|D_\phi(\mathbf{x}_t; t) - D_\theta(\mathbf{x}_t; t)\|$. If KL regularization keeps $d_t < \ell^\Omega(D_\theta(\mathbf{x}_t; t))/L$, then Equation (49) gives

$$\ell^\Omega(D_\phi(\mathbf{x}_t; t)) \ge \ell^\Omega(D_\theta(\mathbf{x}_t; t)) - Ld_t > 0. \tag{51}$$

Therefore, zero violation is impossible inside this neighborhood of the pretrained prediction. $\mathcal{L}_{\text{rollout}}$ might finetune and push $D_\phi$ outside the neighborhood, but $\mathcal{L}_{\text{KL}}$ explicitly penalizes the required movement.

For the fixed EDM reverse-SDE sampler, let $\Delta_k = t_k - t_{k-1} > 0$ and define

$$r_k = \frac{2\Delta_k}{t_k}, \quad \Sigma_k = 2t_k\Delta_k \boldsymbol{I}. \tag{52}$$

Euler-Maruyama gives the discretization step:

$$\mathbf{x}_{k-1} = r_k D_\psi(\mathbf{x}_k; t_k) + (1 - r_k)\mathbf{x}_k + \Sigma_k \mathbf{z}_k, \quad z_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \psi \in \{\theta, \phi\}. \tag{53}$$

Hence the conditional transition has covariance $\Sigma_k$ and mean

$$\mu_\theta(\mathbf{x}_k; t_k) = r_k D_\theta(\mathbf{x}_k; t_k) + (1 - r_k)\mathbf{x}_k, \quad \mu_\phi(\mathbf{x}_k; t_k) = r_k D_\phi(\mathbf{x}_k; t_k) + (1 - r_k)\mathbf{x}_k. \tag{54}$$

So the transition mean difference is

$$\mu_\phi - \mu_\theta = r_k(D_\phi(\mathbf{x}_k; t_k) - D_\theta(\mathbf{x}_k; t_k)), \tag{55}$$

and the stepwise KL is

$$D_{\text{KL},k} = \frac{1}{2} r_k^2 (D_\phi(\mathbf{x}_k; t_k) - D_\theta(\mathbf{x}_k; t_k))^\top \Sigma_k^{-1} (D_\phi(\mathbf{x}_k; t_k) - D_\theta(\mathbf{x}_k; t_k)) \tag{56}$$

$$= \frac{\Delta_k}{t_k^3} (D_\phi(\mathbf{x}_k; t_k) - D_\theta(\mathbf{x}_k; t_k))^\top (D_\phi(\mathbf{x}_k; t_k) - D_\theta(\mathbf{x}_k; t_k)). \tag{57}$$

Combining this with Equation (50), any update that removes the pretrained prediction violation must incur

$$D_{\text{KL,k}} \geq \frac{\Delta_k \left(\ell^\Omega(D_\theta(\mathbf{x}_k; t_k))\right)^2}{t_k^3 L^2}. \tag{58}$$

However, $\ell_{\text{EDM}}$ has a different target: $\mathcal{L}_{\text{EDM}}$ anchors $D_\phi$ to the data rather than to $D_\theta$. Decomposing the EDM error at noise level t as

$$\ell_{\text{EDM}}(\phi) = \mathbb{E}\left[\|D_\phi(\mathbf{x}_t; t) - \mathbb{E}\left[\mathbf{x}_0 \mid \mathbf{x}_t\right]\|^2\right] + \mathbb{E}\left[\text{tr}\left(\text{cov}\left[\mathbf{x}_0 \mid \mathbf{x}_t\right]\right)\right], \tag{59}$$

Equation (46) shows that minimizing $\mathcal{L}_{\text{EDM}}$ reduces the violation bound to $L\sqrt{\mathbb{E}\left[\text{tr}\left(\text{cov}\left[\mathbf{x}_0 \mid \mathbf{x}_t\right]\right)\right]}$. The KL regularizer, by contrast, anchors $D_\phi$ to $D_\theta$ and therefore to its violations. The EDM regularizer alone does not enforce feasibility at intermediate noise levels. It avoids penalizing the corrections that $\mathcal{L}_{\text{rollout}}$ introduces, and the rollout loss supplies the constraint signal. □

## D Multiple constraint gradients and scaling network architecture

The discussion in Section 4.1 considers a single constraint gradient. Further, we apply a symmetric gradient projection inspired by PCGrad (Yu et al., 2020) to handle potential conflicts between multiple gradients. Let $\{\mathbf{g}_j\}_{j=1}^J$ denote the gradients associated with $J$ constraint losses. For each pair of constraints $(\mathbf{g}_j, \mathbf{g}_k)$, if $\mathbf{g}_j^\top \mathbf{g}_k < 0$, we remove the conflicting component from both gradients:

$$\mathbf{g}_k' \leftarrow \mathbf{g}_k - \frac{\mathbf{g}_k^\top \mathbf{g}_j}{\|\mathbf{g}_j\|^2 + \epsilon} \mathbf{g}_j, \tag{60}$$

$$\mathbf{g}_j' \leftarrow \mathbf{g}_j - \frac{\mathbf{g}_j^\top \mathbf{g}_k}{\|\mathbf{g}_k\|^2 + \epsilon} \mathbf{g}_k, \tag{61}$$

where $\epsilon = 10^{-5}$. If $\mathbf{g}_j^\top \mathbf{g}_k \geq 0$, the gradients are left unchanged. The corrected gradients are then combined to form the final constraint direction. Therefore, the scaling network learns a global correction magnitude rather than per-constraint weights while mitigating destructive interference between constraints. In both our experiments, $J = 2$, so this reduces to the symmetric two gradient updates described above.

The scaling network $\gamma_\phi$ is implemented as three MLPs with linear layers and SiLU activations. The first MLP processes the noisy state $\mathbf{x}_t$ to a feature embedding. The second MLP maps the noise level $t$ to a time embedding, which is split into scale and shift components for FiLM-style (Perez et al., 2018) conditioning of the feature embedding. The third MLP maps the conditioned features to $\alpha_\phi$ and $\beta_\phi$. Together, these outputs determine the correction magnitude through Equation (5). Experiment-specific parameterizations of $\alpha_\phi$ and $\beta_\phi$ are provided in the corresponding experiment detail sections.

# E BOUNCING BALLS EXPERIMENT DETAILS

## E.1 BASELINE DETAILS

We compare against a diverse set of baselines spanning unconstrained, sampling-time correction and fine-tuning approaches. EDM (Karras et al., 2022) serves as the unconstrained pretrained diffusion model. MBM (Naderiparizi et al., 2025) and MBM++ (Liang et al., 2026) incorporate constraint gradients into the score function with scaling and guidance functions in Table 3 to enforce constraint satisfaction at the clean sample. MPGD (He et al., 2024) is a sampling-time correction method that we adapt with the same diverging scaling schedule as MBM++, omitting decoder projection. FK-Steering (Singhal et al., 2025) maintains a population of particles and resamples them according to constraint-based rewards within a Sequential Monte Carlo scheme. D-Flow (Ben-Hamu et al., 2024) optimizes the initial sampling noise by backpropagating the terminal constraint violation through the full denoising trajectory and reruns sampling from the updated noise. Both operate on the frozen pretrained EDM without modifying its weights. PIDM (Bastek et al., 2024) adds denoised-state constraint losses to the DSM objective. Since the original formulation does not specify whether pretraining is used, we evaluate both training-from-scratch and fine-tuning variants, denoted PIDM and PIDM (FT). We also include rollout-based fine-tuning baselines, Adjoint Matching (AM) (Domingo-Enrich et al., 2024) and DPOK (Fan et al., 2023), which optimize rollout-level objectives while regularizing deviation from the unconstrained pretrained model. All methods use 50 reverse-time steps.

## E.2 DEFINITION OF CONSTRAINT LOSSES AND EVALUATION METRICS

We provide the full definitions of the constraint losses and evaluation metrics used in the bouncing balls experiment. Each scene contains $B = 10$ balls in a $10 \times 10$ square box, with ball radius $r = 0.5$. Let $\mathbf{x}_\tau^{(b)} \in \mathbb{R}^2$ denote the center position of ball $b$ at frame $\tau$, for $\tau = 1, \cdots, \mathcal{T}$ and $\mathcal{T} = 100$.

**Constraint losses** We first define the boundary and overlap losses used to measure constraint violations. The boundary loss measures ball-wall penetration. Since the center of each ball remains in $[r, 10 - r]^2$, we define

$$\ell_{\text{boundary}}(\mathbf{x}_\tau^{(b)}) = \max\{r - \mathbf{x}_{\tau,1}^{(b)}, \mathbf{x}_{\tau,1}^{(b)} - (10 - r), r - \mathbf{x}_{\tau,2}^{(b)}, \mathbf{x}_{\tau,2}^{(b)} - (10 - r), 0\}. \tag{62}$$

The overlap loss measures overlap between pairs of balls:

$$\ell_{\text{overlap}}(\mathbf{x}_\tau^{(b)}, \mathbf{x}_\tau^{(b')}) = \max(2r - \|\mathbf{x}_\tau^{(b)} - \mathbf{x}_\tau^{(b')}\|, 0). \tag{63}$$

**Reported metrics** The boundary rate is the fraction of frames in which at least one ball has positive boundary loss:

$$\text{Boundary rate} = \frac{1}{\mathcal{T}} \sum_{\tau=1}^{\mathcal{T}} \mathbf{1}\left[\max_b \ell_{\text{boundary}}\left(\mathbf{x}_\tau^{(b)}\right) > 0\right]. \tag{64}$$

The overlap rate is the fraction of frames in which at least one pair of balls has positive overlap loss:

$$\text{Overlap rate} = \frac{1}{\mathcal{T}} \sum_{\tau=1}^{\mathcal{T}} \mathbf{1}\left[\max_{b<b'} \ell_{\text{overlap}}\left(\mathbf{x}_\tau^{(b)}, \mathbf{x}_\tau^{(b')}\right) > 0\right]. \tag{65}$$

The maximum frame-to-frame displacement (F2F) measures abrupt motion or jitter:

$$\text{F2F} = \max_{b,t<\mathcal{T}} \|\mathbf{x}_{\tau+1}^{(b)} - \mathbf{x}_\tau^{(b)}\|_2. \tag{66}$$

The maximum contact distance (MCD) measures whether sharp direction changes are supported by nearby contacts. We estimate velocity directly from consecutive positions, which is more reliable than using predicted velocity in generated samples. Specifically, we define

$$\mathbf{v}_\tau^{(b)} = \mathbf{x}_{\tau+1}^{(b)} - \mathbf{x}_\tau^{(b)}. \tag{67}$$

We then mark $(b, \tau)$ as a candidate bounce if the angle between $\mathbf{v}_{\tau-1}^{(b)}$ and $\mathbf{v}_\tau^{(b)}$ exceeds $\frac{\pi}{6}$. Formally,

$$\mathcal{B} = \left\{ (b,\tau) : \arccos\left( \frac{\mathbf{v}_{\tau-1}^{(b)\top} \mathbf{v}_\tau^{(b)}}{\|\mathbf{v}_{\tau-1}^{(b)}\| \|\mathbf{v}_\tau^{(b)}\| + \epsilon} \right) > \frac{\pi}{6} \right\}, \tag{68}$$

where $\tau = 2, \cdots, \mathcal{T} - 1$. For each candidate bounce $(b, \tau)$, we compute the signed distance to the nearest possible contact object over a local temporal window

$$\mathcal{W}_\tau = \{\tau - 1, \tau, \tau + 1\}. \tag{69}$$

This accounts for small discretization offsets between the detected direction change and the actual contact frame. The signed wall distance is

$$d_{\text{wall}}(b, s) = \min\{\mathbf{x}_{s,1}^{(b)} - r, 10 - r - \mathbf{x}_{s,1}^{(b)}, \mathbf{x}_{s,2}^{(b)} - r, 10 - r - \mathbf{x}_{s,2}^{(b)}\}, \tag{70}$$

and the signed ball-ball distance is

$$d_{\text{ball}}(b, s) = \min_{b' \neq b} \left( \|\mathbf{x}_s^{(b)} - \mathbf{x}_s^{(b')}\| - 2r \right). \tag{71}$$

We collect all candidate signed contact distances in the local window:

$$\mathcal{D}(b, \tau) = \{d_{\text{wall}}(b, s) : s \in \mathcal{W}_\tau\} \cup \{d_{\text{ball}}(b, s) : s \in \mathcal{W}_\tau\}. \tag{72}$$

MCD is then defined as

$$\text{MCD} = \max_{(b,\tau)\in\mathcal{B}} \min_{d\in\mathcal{D}(b,\tau)} |d|. \tag{73}$$

Finally, the maximum energy deviation (MED) measures frame-wise inconsistency in system kinetic energy. Assuming unit mass and constant time step, we define the kinetic energy proxy

$$E_\tau = \sum_{b=1}^{B} \frac{1}{2} \|\mathbf{x}_{\tau+1}^{(b)} - \mathbf{x}_\tau^{(b)}\|_2^2. \tag{74}$$

The maximum energy deviation is:

$$\text{MED} = \max_{t<\mathcal{T}-1} |E_{\tau+1} - E_\tau|. \tag{75}$$

All maximum-based metrics are computed within each generated trajectory and then averaged over evaluation batches. We report the mean and standard deviation over four independent evaluation runs.

### E.3 Ablation study

We conduct ablations to isolate the contribution of each component in our framework. In Table 4, FP training replaces rollout-based training with standard forward process training, which introduces a mismatch between training and sampling dynamics. It keeps the learned scaling network and gradient embeddings trainable. Rollout-only keeps rollout-based training and the same trainable scaling network and gradient embeddings, but removes LoRA adaptations. Finally, IMCD combines rollout-based training approach, the learned scaling network, gradient embeddings, and LoRA adaptation.

We further isolate the effect of the regularizer. In Table 5, KL ablation replaces the EDM denoising loss with the KL regularizer toward the pretrained model. It keeps rollout-based training, the learned scaling network, gradient embeddings, and LoRA adaptation unchanged. Since the pretrained denoiser violates the constraints, anchoring to it can perpetuate these violations, whereas the EDM loss anchors the model to the distribution. KL ablation already outperforms DPOK, indicating that

Table 4: Component of our framework on bouncing ball experiment

| | Constraint (%) ↓ | | Fidelity ↑ | Physical plausibility ($\times 10^{-1}$) ↓ | | |
|---|---|---|---|---|---|---|
| Method | Boundary rate | Overlap rate | r-ELBO ($\times 10^{-2}$) | F2F | MCD | MED |
| FP training | $0.78 \pm 0.03$ | $2.83 \pm 0.06$ | $-21.7 \pm 0.1$ | $4.1 \pm 0.0$ | $0.5 \pm 0.0$ | $1.3 \pm 0.1$ |
| Rollout-only | $0.07 \pm 0.01$ | $0.01 \pm 0.00$ | $-23.4 \pm 0.0$ | $5.4 \pm 0.0$ | $0.9 \pm 0.0$ | $1.9 \pm 0.1$ |
| IMCD | $0.01 \pm 0.00$ | $0.01 \pm 0.00$ | $-22.9 \pm 0.0$ | $3.4 \pm 0.0$ | $1.2 \pm 0.0$ | $0.6 \pm 0.0$ |

Table 5: Regularizer ablation on the bouncing ball experiment.

| | Constraint (%) ↓ | | Fidelity ↑ | Physical plausibility ($\times 10^{-1}$) ↓ | | |
|---|---|---|---|---|---|---|
| Method | Boundary rate | Overlap rate | r-ELBO ($\times 10^{-2}$) | F2F | MCD | MED |
| DPOK | $2.04 \pm 0.05$ | $1.69 \pm 0.03$ | $-53.5 \pm 0.1$ | $2.8 \pm 0.0$ | $4.3 \pm 0.0$ | $0.3 \pm 0.0$ |
| KL ablation | $0.48 \pm 0.02$ | $0.72 \pm 0.01$ | $-23.1 \pm 0.1$ | $3.4 \pm 0.0$ | $0.8 \pm 0.0$ | $0.7 \pm 0.0$ |
| IMCD (Ours) | $0.01 \pm 0.00$ | $0.01 \pm 0.00$ | $-22.9 \pm 0.0$ | $3.4 \pm 0.1$ | $1.2 \pm 0.0$ | $0.6 \pm 0.0$ |

the rollout formulation and architecture account for most of the improvement. Replacing the KL regularizer with the EDM loss further reduces the boundary rate from $0.48\%$ to $0.01\%$ and overlap rate from $0.72\%$ to $0.01\%$, and slightly improves r-ELBO, from $-23.1$ to $-22.9$.

Rather than performing a discrete ablation on the scaling network, we analyze its learned behavior across noise levels. As shown in Figure 3, both learned components are noise dependent: $\alpha_\phi$ is not a constant multiplier, and $\beta_\phi$ varies smoothly along the sampling trajectory. To compare with prior schedules of the form $ct^{-2}$ in Table 3, we plot $t^2\gamma_\phi(\mathbf{x}_t; t)$. This quantity would be constant under such a $ct^{-2}$ schedule, but instead changes substantially across noise levels. This suggests that the scaling network learns a nontrivial modulation beyond a fixed power law. Moreover, $\gamma_\phi(\mathbf{x}_t; t)$ grows rapidly as the noise level decreases, consistent with stronger constraint enforcement near the data manifold.

### E.4 Effect of sampling steps on sampling-time correction

MPGD without projection (He et al., 2024) performs constraint correction only during sampling, without updating the pretrained model. We evaluate it on the bouncing ball task with $50, 100, 150, 200$ denoising steps, where $50$ matches the main method setting. This ablation tests how sensitive sampling-time correction is to the number of steps for incremental updates along the denoising trajectory.

Table 6 shows that MPGD achieves zero boundary and overlap violations across all various sampling steps, but its physical plausibility depends strongly on the number of denoising steps. With $50$ steps, matching the main evaluation setting, MPGD exhibits large frame-to-frame changes (F2F) and high energy deviation (MED), indicating that feasibility is obtained through many incremental corrections. This supports the claim that sampling-time correction methods can enforce constraints, but may rely on many incremental updates to avoid physically infeasible trajectories.

### E.5 Scaling network parameterization

The third MLP outputs $\alpha_\phi$ and a raw exponent parameter $\beta_\phi^{\mathrm{raw}}$. We use a softplus output for $\alpha_\phi$ to ensure a nonnegative magnitude, and map the raw exponent to the effective component as $2\mathrm{sigmoid}(\beta_\phi^{\mathrm{raw}}) - 3$. This offset incorporates the known scaling of the denoised-state guidance term: the score parameterization $(\mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t] - \mathbf{x}_t)/t^2$ contributes an $t^{-2}$ factor, while using the remaining $t^{-1}$ factor as a milder base scaling. Then the network learns the remaining state- and noise-dependent covariance modulation. This parameterization preserves the desired divergence property as noise level approaches zero without requiring a manually tuned schedule.

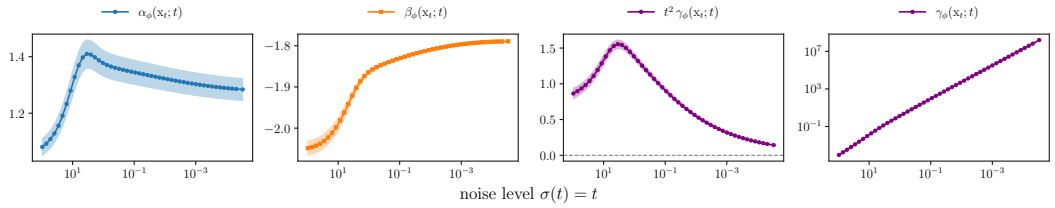



Figure 3: Scaling network output versus noise level $\sigma$ along denoising trajectory for bouncing ball experiment. Means are taken over batches; shaded bands show $\pm 1$ standard error.

Table 6: Effect of sampling steps on MPGD without projection in the bouncing ball task

| | Constraint (%) ↓ | | Physical plausibility ($\times 10^{-1}$) ↓ | | |
|---|---|---|---|---|---|
| Steps | Boundary rate | Overlap rate | F2F | MCD | MED |
| 50 | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $8.6 \pm 0.4$ | $0.8 \pm 0.0$ | $29.7 \pm 5.4$ |
| 100 | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $3.8 \pm 0.2$ | $0.6 \pm 0.0$ | $3.2 \pm 0.9$ |
| 150 | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $3.2 \pm 0.0$ | $0.6 \pm 0.0$ | $0.6 \pm 0.1$ |
| 200 | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $3.1 \pm 0.0$ | $0.6 \pm 0.0$ | $0.4 \pm 0.1$ |

### E.6 Training details

We follow the prior work (Liang et al., 2026) for the experimental setup, using the same transformer-based diffusion backbone (Vaswani et al., 2017; Zhang et al., 2022; Beltagy et al., 2020) and gradient embedding MLP. We use the same log-linear noise schedule for sampling with $\sigma_{\min} = 3 \times 10^{-5}$ and $\sigma_{\max} = 80$ but use 50 sampling steps. The standard EDM model is trained from scratch for 950k iterations with learning rate $3 \times 10^{-4}$ on a Tesla V100 GPU, taking approximately 70 hours. Our proposed fine-tuned model additionally uses the scaling network described in Section D, and includes LoRA adapters. We use gradient checkpointing (Chen et al., 2016) when backpropagating through denoising rollouts to reduce memory consumption. Our model and other baseline models are either initialized from scratch or from the pretrained EDM checkpoint, as specified in their method descriptions, and are trained with learning rate $3 \times 10^{-5}$ for approximately 72 hours on an NVIDIA L40S GPU. The number of training iterations varies across methods due to differences in computational cost. All models use the Adam optimizer; our method and DPOK use batch size 16, while other fine-tuning models use batch size 32.

### E.7 Training-time compute

Because IMCD backpropagates through the full 50-step reverse rollout, its per-iteration cost scales approximately linearly with the rollout length and is higher than that of forward-process fine-tuning methods that optimize a single denoising step. To compare training efficiency under matched compute, we report the number of optimization iterations completed within a 72-hour wall-clock budget on a single NVIDIA L40S GPU in Table 7.

### E.8 Inference-time Overhead

IMCD uses rollout simulation and backpropagation only during training; inference requires neither gradient computation nor test-time optimization. Each reverse step uses two denoiser evaluations: one to compute constraint guidance and one to produce the constraint-aware prediction—resulting in 2 NFE per step. This matches the algorithmic complexity of MBM++ and yields comparable latency (1281 ms versus 1191 ms per sample). Relative to pretrained EDM, IMCD doubles the NFE per step. However, at comparable sample quality (referred to Table 6), IMCD is more efficient than MPGD: MPGD requires 150 steps and 2238 ms per sample, whereas IMCD requires 50 steps and 1281 ms. FK-Steering uses 16 particles and 12 resampling intervals. D-Flow performs 30 test-time optimization rollouts. Effective NFE aggregates evaluations across all particles or rollouts; wall-clock latency additionally captures the backward-pass cost of D-Flow. Thus, IMCD amortizes constraint guidance

Table 7: Number of optimization iterations completed within a matched 72-hour training budget on a single NVIDIA L40S GPU.

| Method | Training iterations |
|---|---|
| MBM | 650K |
| MBM++ | 550K |
| PIDM / PIDM (FT) | 1,300K |
| AM | 0.7K |
| DPOK | 19K |
| IMCD (Ours) | 24K |

Table 8: Inference-time computational cost in the bouncing-ball experiment. NFE denotes the number of denoiser evaluations.

| Method | Steps | NFE/step | Total NFE | Wall-clock (ms/sample) |
|---|---|---|---|---|
| EDM | 50 | 1 | 50 | 548 |
| MPGD | 50 | 1 | 50 | 653 |
| MPGD | 150 | 1 | 150 | 2238 |
| FK-Steering | 50 | 16 | 800 | 1641 |
| D-Flow | 50 | 30 | 1500 | 159846 |
| MBM | 50 | 1 | 50 | 689 |
| MBM++ | 50 | 2 | 100 | 1191 |
| PIDM (FT) | 50 | 1 | 50 | 543 |
| DPOK | 50 | 1 | 50 | 591 |
| AM | 50 | 1 | 50 | 518 |
| IMCD (Ours) | 50 | 2 | 100 | 1281 |

into training and avoids the greater inference-time cost needed by MPGD to attain comparable sample quality. We report the numbers in Table 8.

# F Traffic scene trajectory prediction experiment details

## F.1 Definition of constraint losses and evaluation metrics

We provide the definition of the constraint losses and evaluation metrics used in the traffic scene prediction experiment. Each scene contains a varying number of agents. Given the first 10 frames observed, the model jointly predicts all agents over a future prediction horizon $\mathcal{T} = 30$ frames. We denote the predicted center position of the oriented bounding box for agent $a$ at frame $\tau$ by $\mathbf{x}_\tau^{(a)} \in \mathbb{R}^2$, and denote the corresponding bounding box by $\mathcal{O}_\tau^{(a)}$. In our implementation, the first agent in the predicted scene tensor is the ego agent.

**Constraint losses** The offroad loss measures how far a vehicle extends outside the drivable region. Given the drivable mesh $\mathcal{M}$, we compute the four corners of each vehicle box. For a corner $\mathbf{c}_{\tau,j}^a$, where $j = \{1, \cdots, 4\}$, we define its distance to drivable mesh as

$$d_{\text{mesh}}\left(\mathbf{c}_{\tau,j}^{(a)}\right) = \min_{\mathbf{m}\in\mathcal{M}} \|\mathbf{c}_{\tau,j}^{(a)} - \mathbf{m}\|_2^2. \tag{76}$$

The offroad loss is the maximum distance among corners that lie outside the drivable region:

$$\ell_{\text{offroad}}\left(\mathcal{O}_\tau^{(a)}\right) = \max_j \mathbf{1}\left[\mathbf{c}_{\tau,j}^{(a)} \notin \mathcal{M}\right] d_{\text{mesh}}\left(\mathbf{c}_{\tau,j}^{(a)}\right). \tag{77}$$

The collision loss measures the overlap area between pairs of vehicles. For two agents $a$ and $a'$, we define

$$\ell_{\text{collision}}\left(\mathcal{O}_\tau^{(a)}, \mathcal{O}_\tau^{(a')}\right) = \text{Area}\left(\mathcal{O}_\tau^{(a)}, \mathcal{O}_\tau^{(a')}\right). \tag{78}$$

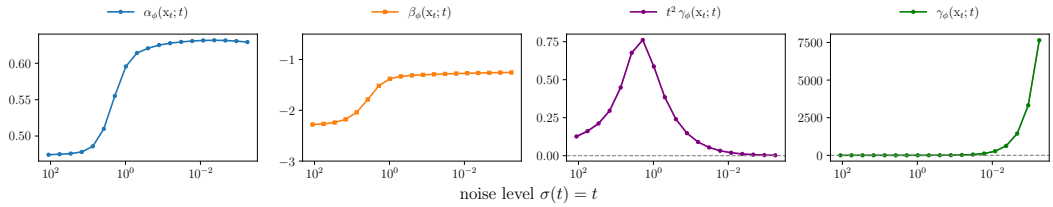



Figure 4: Scaling network output versus noise level $\sigma$ along denoising trajectory for traffic scene trajectory prediction experiment. Means are taken over batches; shaded bands show $\pm 1$ standard error.

**Reported metrics** For each scene, we generate $K = 6$ predicted futures. We use $k \in \{1, \cdots K\}$ to index generated samples and $a = 0$ to denote the ego agent. The offroad rate is the fraction of scenes in which the ego vehicle goes offroad at least once during the future prediction horizon

$$\text{Offroad rate} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}\left[\max_{\tau=1,\cdots,\mathcal{T}} \ell_{\text{offroad}}(\mathcal{O}_\tau^{k,(0)}) > 0\right]. \tag{79}$$

The collision rate is the fraction of scenes in which the ego vehicle collides at least once with another vehicle during the future prediction

$$\text{Collision rate} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}\left[\max_{\tau=1,\cdots,\mathcal{T}} \ell_{\text{collision}}(\mathcal{O}_\tau^{k,(0)}) > 0\right]. \tag{80}$$

The scene maximum frame-to-frame displacement (SF2F) measures abrupt motion or jitter, same as in the bouncing ball experiment:

$$\text{SF2F}_6 = \max_{k,a,\tau<\mathcal{T}} \|\mathbf{x}_{\tau+1}^{k,(a)} - \mathbf{x}_\tau^{k,(a)}\|_2. \tag{81}$$

Both offroad and collision rates are then averaged over evaluation batches.

The scene maximum lateral velocity (SMLV) measures sideways motion relative to each vehicle heading. We estimate velocity from consecutive predicted positions:

$$\mathbf{v}_\tau^{k,(a)} = \mathbf{x}_{\tau+1}^{k,(a)} - \mathbf{x}_\tau^{k,(a)}. \tag{82}$$

Given heading $h_\tau^{k,(a)}$, the lateral velocity is

$$\mathbf{v}_{\text{lat},\tau}^{k,(a)} = -\mathbf{v}_{\tau,x}^{k,(a)} \sin h_\tau^{k,(a)} + \mathbf{v}_{\tau,y}^{k,(a)} \cos h_\tau^{k,(a)}. \tag{83}$$

We define maximum lateral velocity as

$$\text{SMLV}_6 = \max_{k,a,\tau<\mathcal{T}} \|\mathbf{v}_{\text{lat},\tau}^{k,(a)}\|. \tag{84}$$

The maximum final distance (MFD) measures endpoint diversity among generated ego trajectories. We compute the maximum pairwise distance between final predicted ego position across $K$ generated samples,

$$\text{MFD}_6 = \max_{k,k'} \|\mathbf{x}_{\mathcal{T}}^{k,(0)} - \mathbf{x}_{\mathcal{T}}^{k',(0)}\|. \tag{85}$$

$\text{SF2F}_6$, $\text{SMLV}_6$, $\text{MFD}_6$ are computed within $K = 6$ generated samples, and then average over evaluation batches.

### F.2 Method variants

We report two additional variants to our IMCD in Table 9: Rollout-only trains the scaling network and gradient embedding without LoRA adaptation; and IMCD (staged) uses the same trainable components and training objective as IMCD, but initializes from the trained Rollout-only checkpoint before enabling LoRA adapters for a second fine-tuning stage. IMCD is the variant reported in Table 2. As shown in Table 9, IMCD (staged) further reduces the offroad rate, but requires additional training through the initial rollout-only stage.

### F.3 Effect of scaling network and its parameterization

As shown in Figure 4, $\alpha_\phi(\mathbf{x}_t; t)$ and $\beta_\phi(\mathbf{x}_t; t)$ both change with the noise level, so the learned scaling does not collapse to a constant coefficient. Unlike the bouncing balls case, $t^2\gamma_\phi(\mathbf{x}_t; t)$ peaks at an intermediate noise level, suggesting that the model places the strongest relative correction in the middle of the trajectory. The final scale $\gamma_\phi(\mathbf{x}_t; t)$ still grows rapidly as $t$ decreases, preserving strong constraint enforcement near the data manifold.

For parameterization, the third MLP outputs $\alpha_\phi$ and a raw exponent parameter $\beta_\phi^{\text{raw}}$ as before. We apply a softplus to $\alpha_\phi$ to keep the correction magnitude nonnegative, and map the raw exponent to the effective exponent as $2\text{sigmoid}(\beta_\phi^{\text{raw}}) - 3$. This effective exponent, rather than the raw MLP output, is the $\beta_\phi$ used in the scaling function. This aligns with the $-3$ offset in bouncing balls experiment, and the bounded learned exponent keeps the divergence property as $t$ goes to zero.

### F.4 Training details

We follow prior work (Liang et al., 2026) for the experimental setup, using the same transformer-based diffusion backbone (Niedoba et al., 2024) and gradient embedding MLP as the bouncing ball experiment. Our fine-tuned models additionally use the scaling network described in Section D and LoRA adapters. We use the same log-linear sampling noise schedule with $\sigma_{\max} = 80$ and $\sigma_{\min} = 2 \times 10^{-4}$, and sample with 20 steps. The standard diffusion model is trained from scratch with learning rate $3 \times 10^{-4}$ on a Tesla V100 GPU, taking approximately 168 hours for about 140 epochs. Our fine-tuned models, and other fine-tuned baselines are trained with learning rate $3 \times 10^{-4}$ for approximately 72 hours on two NVIDIA L40S GPUs, except for PIDM, which uses learning rate $3 \times 10^{-5}$, and IMCD (staged). The IMCD (staged) variant follows the staged procedure described above: the Rollout-only model is trained for three days, and the LoRA adapters are then trained for another three days. All methods sample with 20 steps, matching the main evaluation. We additionally evaluate MPGD with 200 sampling steps, since sampling-time correction depends strongly on the step count shown in Section E.4, and report it as MPGD (200 steps). The number of training iterations varies across methods due to differences in computational cost. All models use the Adam optimizer with batch size 16.

### F.5 More experimental results

Table 9: Traffic scene trajectory prediction results on the INTERACTION DR_DEU_Merging_MT scenario.

| | Constraint (%) ↓ | | Min traj. match ↓ | | | | Phys. plaus. ↓ | | Diversity ↑ |
|---|---|---|---|---|---|---|---|---|---|
| Method | Offroad | Collision | $ADE_6$ | $FDE_6$ | $SADE_6$ | $SFDE_6$ | $SF2F_6$ | $SMLV_6$ | $MFD_6$ |
| MPGD (200 steps) | 0.29 | 0.32 | 0.19 | 0.47 | 0.25 | 0.71 | 1.07 | 0.38 | 2.30 |
| Rollout-only | 0.49 | 0.02 | 0.18 | 0.45 | 0.23 | 0.66 | 1.10 | 0.49 | 1.92 |
| IMCD | 0.30 | 0.00 | 0.18 | 0.45 | 0.23 | 0.66 | 1.07 | 0.43 | 1.91 |
| IMCD (staged) | 0.04 | 0.00 | 0.19 | 0.47 | 0.24 | 0.69 | 1.02 | 0.37 | 1.96 |

In the rendered visualizations below, we show the final predicted frame so that the full trajectory is visible. Gray dot trajectories indicate ground-truth agent motion, and orange trajectories denote model predictions.

Table 10: Traffic scene trajectory prediction results on the INTERACTION DR_DEU_Roundabout_OF scenario.

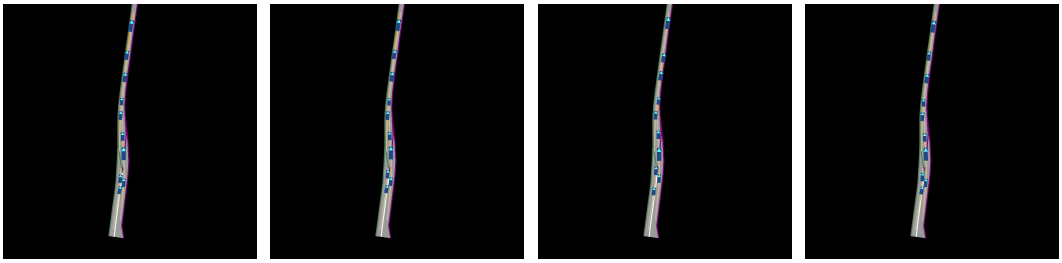
| | Constraint (%) ↓ | | Min traj. match ↓ | | | | Phys. plaus. ↓ | | Diversity ↑ |
|---|---|---|---|---|---|---|---|---|---|
| Method | Offroad | Collision | $ADE_6$ | $FDE_6$ | $SADE_6$ | $SFDE_6$ | $SF2F_6$ | $SMLV_6$ | $MFD_6$ |
| EDM | 23.92 | 1.70 | 0.35 | 1.11 | 0.50 | 1.80 | 1.15 | 0.49 | 6.00 |
| MPGD | 11.96 | 1.42 | 0.33 | 1.04 | 0.45 | 1.57 | 1.24 | 0.65 | 3.98 |
| MPGD (200 steps) | 1.78 | 1.62 | 0.34 | 1.00 | 0.48 | 1.63 | 1.27 | 0.66 | 4.85 |
| MBM++ | 12.42 | 1.40 | 0.32 | 1.05 | 0.44 | 1.59 | 1.21 | 0.61 | 3.91 |
| PIDM (FT) | 14.12 | 1.35 | 0.34 | 1.24 | 0.49 | 1.93 | 0.96 | 0.33 | 5.00 |
| DPOK | 3.73 | 0.11 | 2.99 | 8.88 | 2.96 | 9.94 | 0.97 | 0.56 | 2.42 |
| CriticSMC | – | 0.08 | 0.45 | – | – | – | – | – | 3.43 |
| Rollout-only | 0.68 | 0.03 | 0.37 | 1.06 | 0.51 | 1.63 | 2.07 | 1.35 | 4.07 |
| IMCD | 1.22 | 0.01 | 0.37 | 1.07 | 0.51 | 1.66 | 1.69 | 1.06 | 4.02 |
| IMCD (staged) | 0.37 | 0.00 | 0.37 | 1.08 | 0.51 | 1.66 | 1.68 | 1.05 | 3.95 |
| Ground truth | 0.00 | 0.00 | – | – | – | – | 0.84 | 0.06 | – |

Table 11: Traffic scene trajectory prediction results on the INTERACTION DR_USA_Intersection_MA

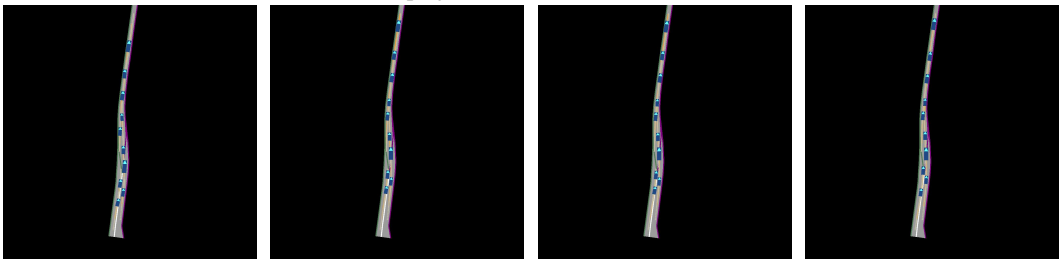
| | Constraint (%) ↓ | | Min traj. match ↓ | | | | Phys. plaus. ↓ | | Diversity ↑ |
|---|---|---|---|---|---|---|---|---|---|
| Method | Offroad | Collision | $ADE_6$ | $FDE_6$ | $SADE_6$ | $SFDE_6$ | $SF2F_6$ | $SMLV_6$ | $MFD_6$ |
| EDM | 1.36 | 1.54 | 0.24 | 0.66 | 0.40 | 1.37 | 1.29 | 0.46 | 4.44 |
| MPGD | 0.49 | 0.83 | 0.22 | 0.64 | 0.35 | 1.22 | 1.24 | 0.42 | 3.17 |
| MPGD (200 steps) | 0.04 | 1.13 | 0.23 | 0.63 | 0.37 | 1.28 | 1.26 | 0.45 | 3.91 |
| MBM++ | 0.42 | 0.83 | 0.20 | 0.63 | 0.33 | 1.21 | 1.22 | 0.36 | 3.11 |
| PIDM (FT) | 1.15 | 1.20 | 0.21 | 0.67 | 0.36 | 1.39 | 1.20 | 0.28 | 4.26 |
| DPOK | 0.37 | 0.04 | 2.07 | 6.28 | 2.10 | 6.60 | 0.96 | 0.52 | 1.77 |
| CriticSMC | – | 0.09 | 0.45 | – | – | – | – | – | 2.87 |
| Rollout-only | 0.03 | 0.00 | 0.23 | 0.64 | 0.35 | 1.22 | 1.33 | 0.53 | 3.17 |
| IMCD | 0.05 | 0.00 | 0.22 | 0.65 | 0.35 | 1.23 | 1.26 | 0.46 | 3.17 |
| IMCD (staged) | 0.01 | 0.00 | 0.23 | 0.67 | 0.36 | 1.26 | 1.24 | 0.45 | 3.17 |
| Ground truth | 0.00 | 0.00 | – | – | – | – | 1.11 | 0.05 | – |

Table 12: Traffic scene trajectory prediction results on the INTERACTION DR_USA_Roundabout_FT

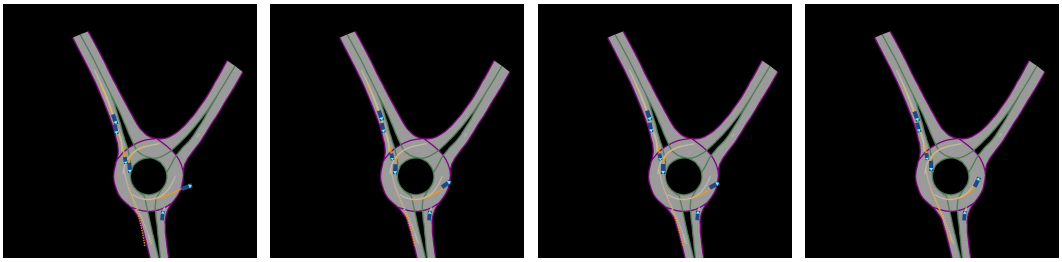
| | Constraint (%) ↓ | | Min traj. match ↓ | | | | Phys. plaus. ↓ | | Diversity ↑ |
|---|---|---|---|---|---|---|---|---|---|
| Method | Offroad | Collision | $ADE_6$ | $FDE_6$ | $SADE_6$ | $SFDE_6$ | $F2F_6$ | $MLV_6$ | $MFD_6$ |
| EDM | 9.32 | 1.37 | 0.25 | 0.72 | 0.41 | 1.44 | 1.23 | 0.47 | 4.26 |
| MPGD | 2.64 | 0.93 | 0.23 | 0.68 | 0.35 | 1.24 | 1.21 | 0.50 | 2.97 |
| MPGD (200 steps) | 0.26 | 1.13 | 0.23 | 0.68 | 0.37 | 1.32 | 1.24 | 0.52 | 3.63 |
| MBM++ | 2.49 | 0.92 | 0.21 | 0.68 | 0.34 | 1.23 | 1.18 | 0.46 | 2.92 |
| PIDM (FT) | 5.79 | 0.86 | 0.23 | 0.83 | 0.39 | 1.53 | 1.02 | 0.31 | 3.87 |
| DPOK | 1.95 | 0.06 | 2.55 | 7.84 | 2.65 | 8.25 | 1.04 | 0.58 | 1.95 |
| CriticSMC | – | 0.07 | 0.44 | – | – | – | – | – | 2.97 |
| Rollout-only | 0.28 | 0.11 | 0.24 | 0.70 | 0.36 | 1.26 | 1.60 | 1.01 | 3.01 |
| IMCD | 0.25 | 0.06 | 0.24 | 0.71 | 0.37 | 1.28 | 1.37 | 0.75 | 2.94 |
| IMCD (staged) | 0.16 | 0.05 | 0.24 | 0.72 | 0.36 | 1.29 | 1.35 | 0.77 | 3.00 |
| Ground truth | 0.00 | 0.00 | – | – | – | – | 0.95 | 0.06 | – |



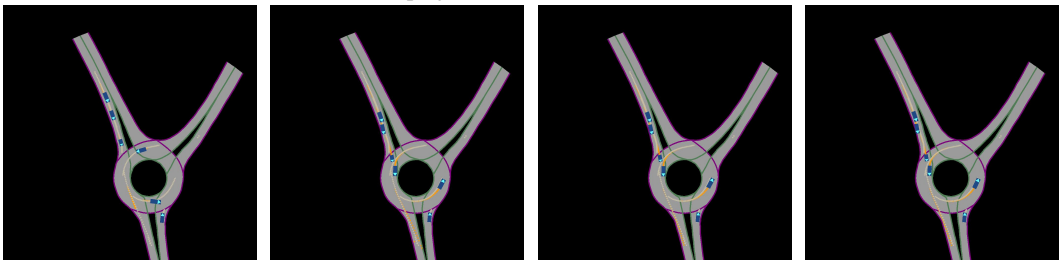
Standard Diffusion | MPGD w/o projection | MBM++ | PIDM



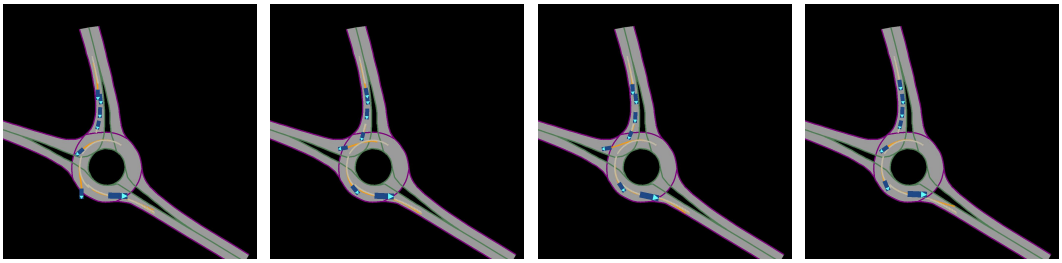
DPOK | Rollout-only | IMCD | IMCD (staged)



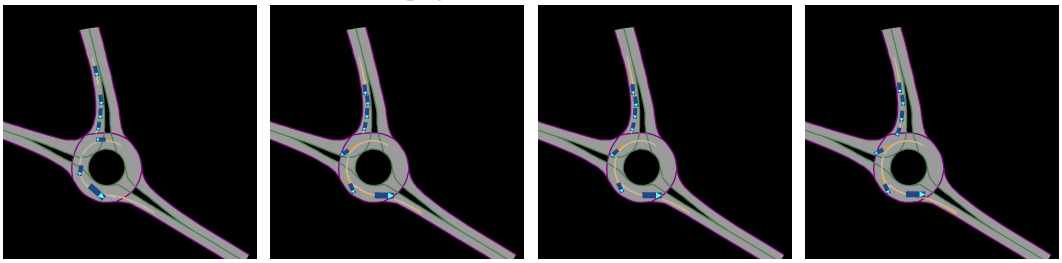
Standard Diffusion | MPGD w/o projection | MBM++ | PIDM



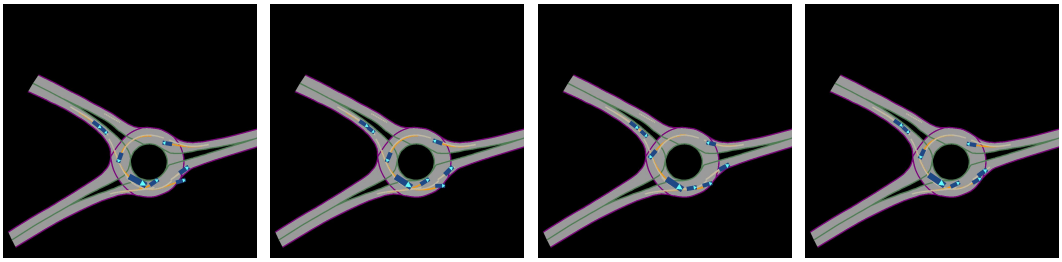
DPOK | Rollout-only | IMCD | IMCD (staged)



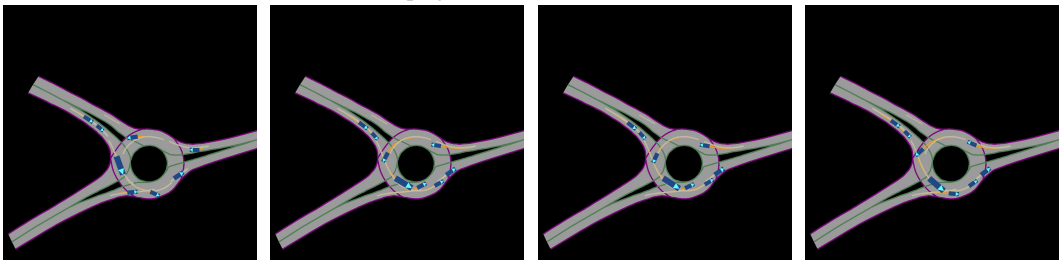
Standard Diffusion | MPGD w/o projection | MBM++ | PIDM



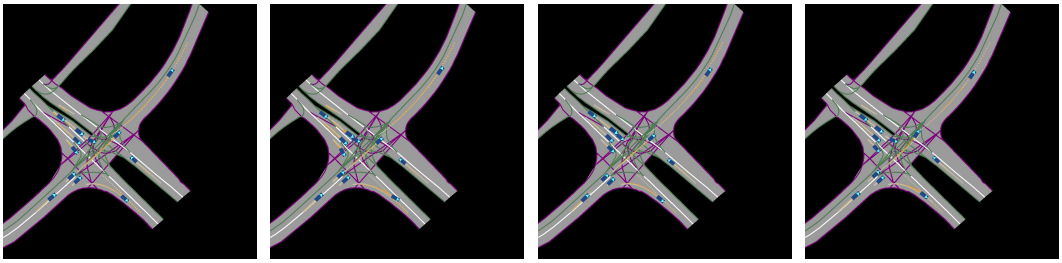
DPOK | Rollout-only | IMCD | IMCD (staged)



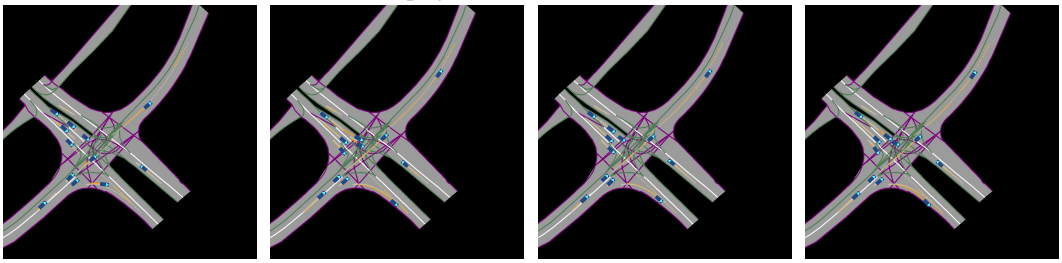
Standard Diffusion | MPGD w/o projection | MBM++ | PIDM



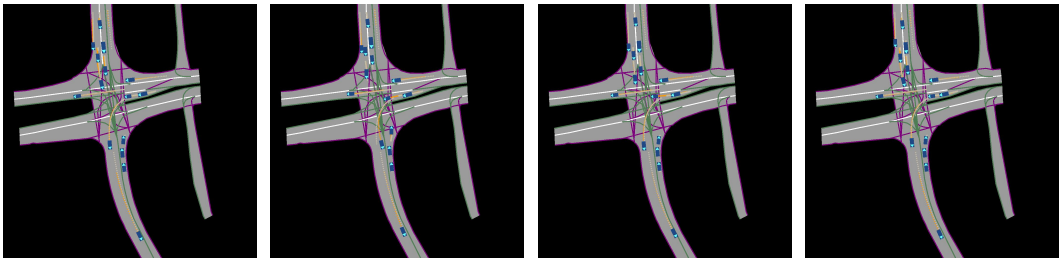
DPOK | Rollout-only | IMCD | IMCD (staged)



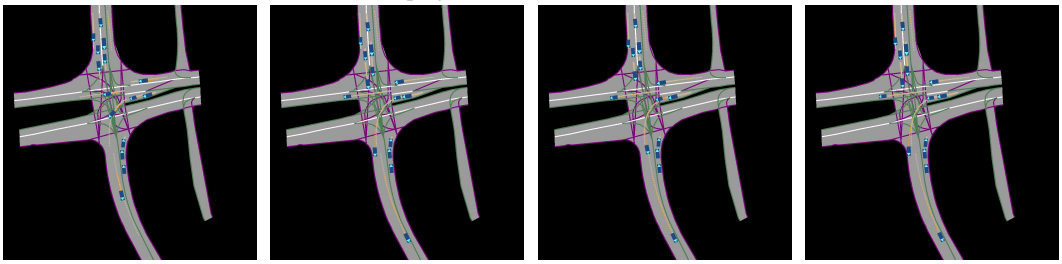
Standard Diffusion | MPGD w/o projection | MBM++ | PIDM



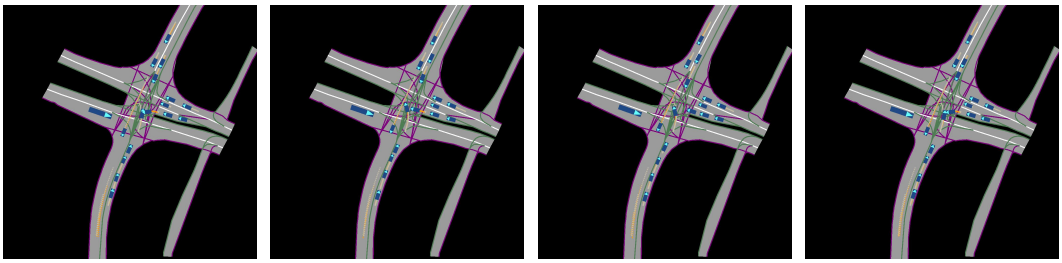
DPOK | Rollout-only | IMCD | IMCD (staged)



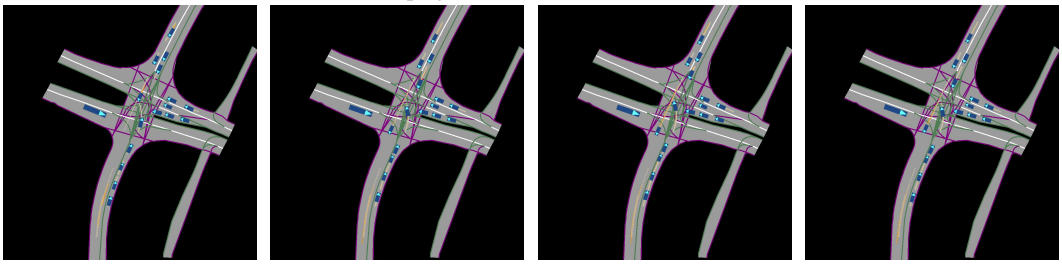
Standard Diffusion | MPGD w/o projection | MBM++ | PIDM



DPOK | Rollout-only | IMCD | IMCD (staged)



Standard Diffusion | MPGD w/o projection | MBM++ | PIDM



DPOK | Rollout-only | IMCD | IMCD (staged)