KURAMOTO DECISION DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion-based decision making has seen explosive popularity due to the ability in producing rich, multimodal trajectories. However, existing frameworks erase the structure of robotic data, which is critical for action fluidity and non-myopic planning, through stochastic noising, then denoise or recover it only probabilistically. This paper aims to address the problem of incorporating spatio-temporal dependencies into the diffusion processes (i.e., forward and reverse), without sacrificing trajectory quality. Building upon and improving the work of Song et al., we propose **KDDM**, a physics-inspired diffusion paradigm that explicitly encodes the information through Kuramoto-based synchronization. By two neat tricks to harmonize the intertwined chains of Langevin and Kuramoto dynamics, we prove that **KDDM** elegantly converges to the desired outcome and theoretically outperforms prior approaches. Furthermore, we extend **KDDM** to ODE-accelerated and guided sampling, enabling its integration with today's decision-making ecosystem. Experimentally, our proposals achieve superior performance in general, across an extensive range of tasks, including trajectory augmentation, long-horizon planning, and (real-world) robotic control, compared to various diffusion-based solutions.

Website: https://kuramoto-decision-diffusion.github.io

1 INTRODUCTION

The dexterity of a craftsman, the grace of a dancer, and the harmony of a flock in flight. Those are only possible when robots are endowed with a decision-making "brain" that produces instructions extending beyond individual moments and components (e.g., arm joints and swarm members). In a string of recent papers (Wolf et al., 2025; Song et al., 2025), to realize such fluidity with machine learning (ML) techniques, perhaps the best practice comes from diffusion-based generative robotic control/planning/augmentation, including Diffuser (Janner et al., 2022), Diffusion Policy (Chi et al., 2025), and the action expert in π-series Vision-Language-Action (VLA) models (Black et al., 2024; Intelligence et al., 2025b;a; 2026). The basic idea is to learn a diffusion process that jointly generates a subsequence of states and/or actions, named trajectory chunk, so that future in-distribution decisions are not tied to single-step rewards alone. But notice that, in theory, the formulation of classical diffusions, ranging from Denoising Diffusion Probabilistic Model (DDPM, Ho et al., 2020) to Rectified Flow (R-Flow, Liu et al., 2023), is *point-wise* since it predefines that the (de)noising operation of each action/state is mutually *independent*. As a result, the burden of ensuring global consistency can only be shouldered by the underlying neural network (NN), and these black-box AI systems aren't always so lucky, which motivates one to ask: To what extent, and how, can the diffusion framework itself characterize *spatio-temporal dependencies* in decision-making tasks?

Most recently, Song et al. (2026) gave us inspirations to tackle this problem from a new direction called *Kuramoto model* (Kuramoto, 1984), a mathematical model that speaks to the very nature of synchronization phenomena in neural and physical systems. Specifically, they developed Kuramoto Orientation Diffusion Models (KODMs) for structured data generation, in which target points $\mathbf{x}$ are viewed as *coupled oscillators* $\boldsymbol{\theta}$, while the forward and reverse processes perform *synchronization* and *desynchronization* respectively, by leveraging stochastic Kuramoto dynamics $\boldsymbol{\theta}_{0\to T}$. Here at an intuitive level, chunked trajectories should also be orientation-rich. However, our test (§3) suggests that simply transplanting the scheme as-is often fares much worse than classical approaches in decision-making scenarios. The trouble is that the (de)synchronization can neither be rewound nor jumped ahead, due to its non-linear operations (e.g., phase wrapping) on periodic domains, i.e., $\boldsymbol{\theta}_t \in [-\pi, \pi]$. This is more than a matter of efficiency — they indeed compromise the variational lower bound and rule out prevalent optimized sampling algorithms (Ye et al., 2024; Lai et al., 2025).

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.

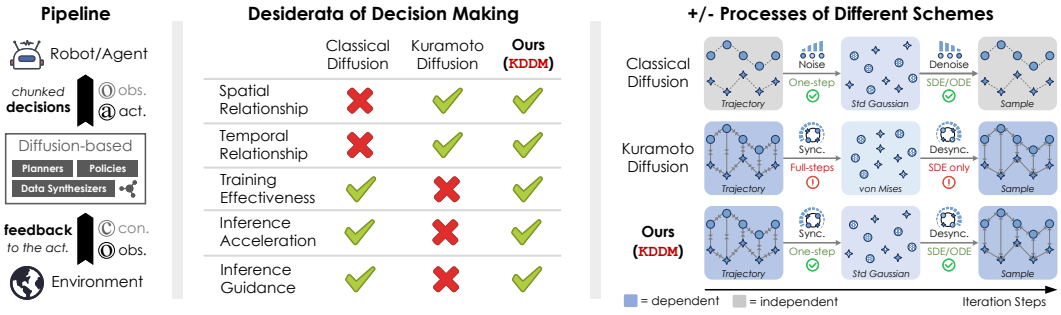


Figure 1: **Left:** Pipeline and roles of diffusion models in robotic decision-making scenarios. **Middle:** Comparison of diffusion backbone classes with respect to 5 desiderata. **Right:** Behavior of different diffusion schemes as forward (+) and reverse (-) processes.

To address these flaws, we introduce *Kuramoto Decision Diffusion Model* (**KDDM**), a novel diffusion paradigm tailored for coherent decision-making, which strikes a sweet spot between classical and kuramoto diffusion (see Fig. 1, middle & right). Our key insight is to peel away non-linear parts from KODM until all that remains is the predictable interaction between oscillators + DDPM. Two core optimizations in **KDDM** are devised as follows. ❶ **Self-Synchronization**: We instead calculate Kuramoto interactions $\mathbf{K}_t$, i.e., coupling terms for pulling similar points at step t together, directly from $\boldsymbol{\theta}_0$ independent of $\boldsymbol{\theta}_{1:T}$. ❷ **Telescoping Update**: We design a simple yet effective transition kernel $\boldsymbol{\theta}_{t-1} \to \boldsymbol{\theta}_t$, in which new interactions make way by clearing the old. In doing so, we prove that the diffusion process in **KDDM** elegantly converges to tractable marginal distributions that allow spatio-temporal relationships to be woven into the decision-making system without trade-offs. What's more, the training of **KDDM** is easy and stable, with a theoretically better bound on log-likelihood, and we extend **KDDM**'s sampling procedure to seamlessly integrate with existing frameworks: **(i)** we present its ordinary differential equation (ODE) version, *Kuramoto Decision Implicit Model* (**KDIM**), which enables considerable acceleration compared to stochastic differential equations (SDEs); and **(ii)** we unlock its full potential for controllable decision-making tasks by deriving a suite of conditioning strategies, such as sequential monte carlo, training-free and classifier(-free) guidance.

Through extensive experiments in multiple Reinforcement Learning (RL) and Imitation Learning (IL) environments for over 10 algorithms and their variants, we find general performance gains across 3 categories of decision-making roles: synthesizers, planners, and policies. It shows potential for up to 16% data quality (e.g., smoothness and realness) and $4\times$ success rate improvement, by plugging **KDDM** or **KDIM** into off-the-shelf diffusion-based solutions. We also demonstrate their effectiveness on real-world robot manipulation tasks. ***Please refer to this URL for the robotic arm videos.***

2 Preliminary

Problem Statement. In this paper, we focus on the sequential decision-making problem using trajectory optimization (Tassa et al., 2012). Consider a robotic system governed by discrete-time dynamics $\mathbf{s}_{t+1} = \boldsymbol{d}(\mathbf{s}_t, \mathbf{a}_t)$ at state $\mathbf{s}_t$ given an action $\mathbf{a}_t$, then we want to find a *sequence* of actions $\mathbf{a}^*_{0:H}$, where H is the chunk size, to maximize cumulative reward or mimic the expert behaviors closely. We assume access to an offline dataset of near-optimal trajectories $\mathcal{D} = \{(\boldsymbol{\tau}, \mathbf{c})_j\}$, where each chunked trajectory $\boldsymbol{\tau} = ([\mathbf{s}_0, \mathbf{a}_0], [\mathbf{s}_1, \mathbf{a}_1], \ldots, [\mathbf{s}_H, \mathbf{a}_H])$, and the condition $\mathbf{c}$ can be a variety of additional inputs to the model, e.g., RGB images, goal positions, and other task descriptors.

Kuramoto Model. Like Winfree (1967), Kuramoto model (Kuramoto, 1984) is a non-linear dynamical system of phase oscillators $\{\theta^i\}_{i=1}^N$ with specified natural frequencies $\{\omega_i\}_{i=1}^N$. If the coupling is strong enough, the system will evolve to one with all oscillators in phase lock (Ermentrout, 1985; Van Hemmen & Wreszinski, 1993), which is called *synchronization* as illustrated in Fig. 2. Such evolution is dictated by the governing equations $\dot{\theta}^i = \omega_i + \frac{1}{N}\sum_{j=1}^N \sin(\theta^j - \theta^i)$, and it further introduces the order parameter $r(t)e^{i\psi(t)} = \frac{1}{N}\sum_{j=1}^N e^{i\theta_t^j}$, where ψ is the same average phase, in order to quantify the overall synchrony with $r \in [0, 1]$, i.e., the higher the greater synchrony.

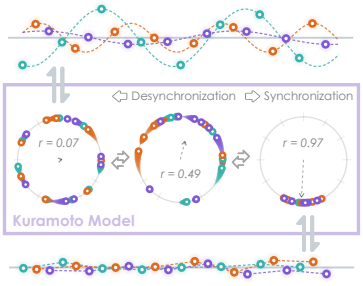
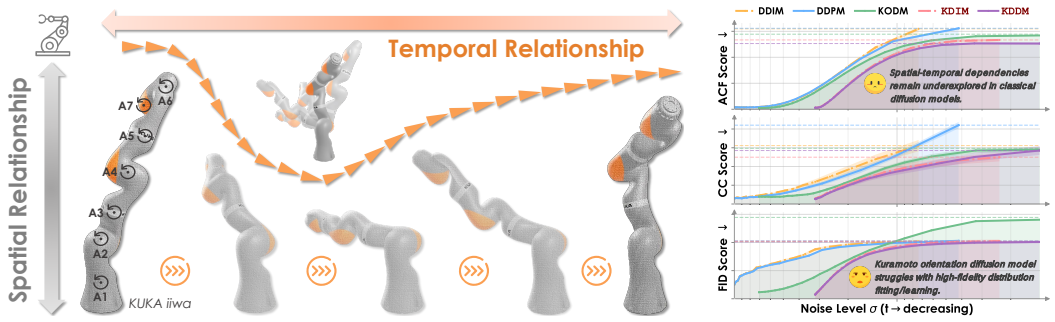


Figure 2: The Kuramoto model.



(a) Simple illustration using a 7-DOF robotic arm (KUKA iiwa) (b) Gen. data quality on MuJoCo

Figure 3: Motivating analysis — **(a)** toy example that highlights the importance of spatio-temporal dependencies; and **(b)** empirical comparisons using the simulated data from MuJoCo environments.

Kuramoto Diffusion. Diffusion models (DMs, Sohl-Dickstein et al., 2015; Ho et al., 2020) consist of the forward and reverse processes that can be formulated as a pair of SDEs. As the Kuramoto model admits a similar Markov-chain form, KODM (Song et al., 2026) intertwines the two through Langevin dynamics (Parisi, 1981; Grenander & Miller, 1994), by the following forward SDE:

$$\frac{d\theta_t^i}{dt} = \frac{1}{N}\sum_{j=1}^{N} \kappa_t \sin(\theta_t^j - \theta_t^i) + \kappa_t^{\text{ref}} \sin(\psi_{\text{ref}} - \theta_t^i) + \sigma_t \boldsymbol{\epsilon}^i, \quad \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{1}$$

where κ_t is the time-varying coupling strength, κ_t^{ref} the reference-phase coupling strength, and σ_t the noise level. Hereafter, let $\boldsymbol{\theta}_t := \{\theta_t^i\}_{i=1}^N = (\boldsymbol{\theta}_t + \pi) \bmod (2\pi) - \pi$. Given $\boldsymbol{\theta}_T$ sampled from a *von Mises* distribution under *quasi-equilibrium*, the corresponding time-reversed SDE is constructed as

$$\frac{d\theta_t^i}{dt} = \frac{1}{N}\sum_{j=1}^{N} \kappa_t \sin(\theta_t^j - \theta_t^i) + \kappa_t^{\text{ref}} \sin(\psi_{\text{ref}} - \theta_t^i) - \sigma_t^2 \frac{\partial}{\partial \theta_t^i} \log p(\boldsymbol{\theta}_t) + \sigma_t \bar{\boldsymbol{\epsilon}}^i, \quad \bar{\boldsymbol{\epsilon}}^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2}$$

Since the marginal density $p(\boldsymbol{\theta}_t)$ cannot be obtained directly, the training of KODM is achieved by *Local Score Matching* (LSM, Vincent, 2011), whose objective can be written as

$$\mathbb{E}_{\boldsymbol{\theta}_0} \mathbb{E}_{t\sim U[0,T]} \mathbb{E}_{\boldsymbol{\theta}_{t-1}\sim p(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0)} \mathbb{E}_{\boldsymbol{\theta}_t \sim p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})} \left[\sigma_t^2 \left\| \mathbf{s}\left(\boldsymbol{\theta}_t, t\right) - \nabla \log p\left(\boldsymbol{\theta}_t | \boldsymbol{\theta}_{t-1}\right) \right\|_2^2 \right]. \tag{3}$$

3 Motivation

Revisiting DMs for decision making. Looking across the vibrant landscape of decision-making with DMs (Dong et al., 2024b), current studies trace 3 lines (also see Fig. 1, left): generating long-horizon trajectories and executing them as *planners* (Janner et al., 2022; Ajay et al., 2023; Lu et al., 2025b); replacing conventional Gaussian policies with (multimodal) diffusion *policies* (Pearce et al., 2023; Chi et al., 2025); and serving as data *synthesizers* to enrich the training data (Lu et al., 2023). Whether conditional or unconditional generation, they gradually inject random Gaussian noise into trajectories $\boldsymbol{\tau}$, i.e., $\mathbf{x}_0$, to create $\{\mathbf{x}_t\}_{t=1}^T$ in the forward process, then the reverse process transforms samples from a prior distribution $p(\mathbf{x}_T)$ by utilizing numerical solvers for the SDE/ODE. Popular DMs includes DDPM (Ho et al., 2020), DDIM (Song et al., 2020a), EDM (Karras et al., 2022), DPM-Solver (Lu et al., 2022; 2025a), and R-Flow (Liu et al., 2023); see Appendix A and C.2 for more details.

Where might these DMs fall short? It has become common to design the systems in the mold of sequence or chunk modeling to achieve effective decision-making (Chen et al., 2021). To get a feel, consider the illustrative example in Fig. 3(a). For this 7-DOF robotic arm to move coherently, one must account for both its *spatial relationship* (how all joints collaborate) and *temporal relationship* (how motions unfold over time). Then, the Achilles' Heel of the above DMs begins to surface: *They completely ruin the structure of chunked trajectories with independent noise, while sampling and refining each state/action in isolation, theoretically blind to spatio-temporal dependencies.* Although such a predicament might be somewhat alleviated by employing a seq-to-seq model (Chen et al., 2024a; Yoon et al., 2025), this is a parallel, non-conflicting line to our work. Empirically, we tested the sample quality of classical

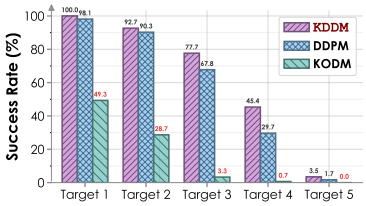


Figure 4: Success rates of Diffusion BC in Relay Kitchen.

DMs based on Transformer in the MuJoCo environment. As can be seen in Fig. 3(b), where ACF and CC score (definitions in Appendix C.3) are metrics that measure how well the generated data capture temporal and spatial correlations of the real data respectively, there is indeed room for improvement.

Can Kuramoto answer the call? Naturally, the Kuramoto diffusion (i.e., KODM) stands out as a compelling candidate here, which encourages local patterns and orientations to reinforce one another before noise erodes these features. Given data $\boldsymbol{\tau}$ mapped to angular phase variables $\boldsymbol{\theta}$ in $[-\pi, \pi]$, it can progressively (de)synchronize all points (e.g., velocities and positions) as in Eqns. (1) and (2), thereby capturing spatio-temporal dependencies of interest. Nevertheless, this straight-off-the-shelf method is barely capable of handling our decision-making tasks. A primary limitation lies in the *efficiency*: each forward process incurs an $\mathcal{O}(T)$ time cost, and soft sampling acceleration (i.e., ODE) is unavailable. More importantly, the training objective in Eqn. (3) is *sub-optimal*, leading to a poorly learned core function $\mathbf{s}(\boldsymbol{\theta}_t, t)$ that may produce unreal or failed trajectories (see Fig. 3(b), 3rd row). Empirically, replacing the backbone of Diffusion Behavior Cloning (BC) with KODM, Fig. 4 shows a significant degradation in robotic manipulation performance in the Relay Kitchen environment.

Summary. Our motivating analysis reveals a dilemma: *We seek the dependency-capturing power of KODM without giving up the capabilities of conventional DMs for decision making.* This could be fiendishly tricky, because any (even slight) interaction between points would make things a lot more complicated, preventing the diffusion process from converging to a stationary distribution.

4 Methodology

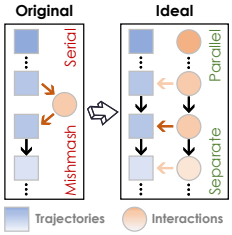


Figure 5: Our insight.

In this section, we establish that there is, however, a way out. We introduce **KDDM** and its extensions to embrace the best of both worlds, i.e., preserved spatio-temporal dependencies and high log-likelihoods, outperforming other DMs under decision-making conditions with theoretical derivations. Before delving into the details, we describe the geometric insight behind **KDDM**.

Intuition: Because of the presence of the non-linear drift in Eqns. (1) and (2), we cannot explicitly write down a *one-step* forward mapping $\boldsymbol{\theta}_0 \to \boldsymbol{\theta}_t$, such as $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ in DDPMs where $\alpha_t := 1 - \beta_t \in (0, 1]$, and consequently the score function is also uncomputable in a closed form. More pointedly, we observe that interleaving Kuramoto coupling with noise injection results in a lose-lose situation: the former disrupts the desired linear property of Gaussians, while the latter destabilizes the synchronization started from $\boldsymbol{\theta}_0$. To this end, we aim to disentangle them into two parallel chains with minimal overlap (Fig. 5), a separation that, intuitively, makes the diffusion process more tractable.

4.1 Two Simple Tweaks ⇒ a Turnaround

Guided by the insights above, **KDDM** employs two main optimizations over KODM, stated as follows.

1 Self-Synchronization. First of all, in designing **KDDM**, we seek to simplify the non-linear drift to better support separable (de)synchronization. Instead of treating the coupling among oscillators in $\boldsymbol{\theta}_t$ as an attractor, **KDDM** targets *noiseless* interactions. By Lemma 5, we can write the t-coupling term as

$$\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right) := r(t)\sin\left(\psi(t) - \boxed{\boldsymbol{\theta}_0(t)}\right), \quad \text{where } \boldsymbol{\theta}_0(t) = \boldsymbol{\theta}_0(t-1) + \mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t-1\right).$$

Here the ψ_{ref}-reference-phase coupling is also neglected. Notably, once $\boldsymbol{\theta}_0$ is specified, $\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)$ is uniquely determined for any t. In other words, $\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)$ is predictable, and hence learnable.

2 Telescoping Update. Now, owing to the cumulative effect of interactions, i.e., $\sum_{i=1}^{t}\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, i\right)$, there is still no shortcut to be found, with the two chains remaining largely overlapped. **KDDM** adopts a neat trick known as *telescoping* to address the challenges. At its core, a new forward transition kernel is defined by Eqn. (4), viewed from the "Variance-Preserving" perspective:

$$q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_0\right) = \mathcal{N}\left(\sqrt{\alpha_t}\boldsymbol{\theta}_{t-1} + \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)\boxed{-\sqrt{\alpha_t}\kappa_{t-1}\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t-1\right)}, \beta_t\mathbf{I}\right). \tag{4}$$

The equality means that whenever a new coupling is introduced, **KDDM** clears the previously accumulated synchronization "cache". To picture this, imagine a broken zipper being pulled from top to bottom (and vice versa): the two sides remain separated above and below the slider, and are joined only around its position. Such a design choice ensures that at each diffusion timestep, **KDDM** adds a transient drift that is explicitly biased by the spatio-temporal dependency, without disturbing our desired closed-form properties. In particular, we can directly write the resulting marginal density as

$$q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right), \left(1-\bar{\alpha}_t\right)\mathbf{I}\right). \tag{5}$$

Algorithm 1 Training of **KDDM**

1: **repeat**
2: Draw $\boldsymbol{\theta}_0 \sim q(\boldsymbol{\theta}_0)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: Sample $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\mathbf{K}_t \leftarrow \text{SELF-SYNCHRONIZATION}(\boldsymbol{\theta}_0, t)$
5: $\boldsymbol{\theta}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \mathbf{K}_t + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
6: Update NN param. with $\nabla \|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0(\boldsymbol{\theta}_t, t)\|^2$
7: **until** converged

Algorithm 2 Sampling of **KDDM**

1: Draw $\boldsymbol{\theta}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\hat{\mathbf{K}}_t \leftarrow \text{SELF-SYNCHRONIZATION}(\hat{\boldsymbol{\theta}}_0, t)$
5: $\boldsymbol{\theta}_{t-1} = \tilde{\boldsymbol{\mu}}(\boldsymbol{\theta}_t, \hat{\boldsymbol{\theta}}_0, \hat{\mathbf{K}}_t) + \sigma_t \mathbf{z}$ ▷ Eqn. (6)
6: **end for**
7: **return** $\boldsymbol{\theta}_0$

To that end, notice that since the coupling term $\mathbf{K}_\psi(\boldsymbol{\theta}_0, T)$ decreases towards near-zero for sufficiently large T (see Lemma 7), the stationary distribution $p(\boldsymbol{\theta}_T)$ further simplifies to a well-known isotropic Gaussian form, i.e., $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Corresponding to Eqs. (4) and (5), the reverse, or desynchronization process of **KDDM**, can then be formulated with Gaussian distributions conditioned on $\boldsymbol{\theta}_0$ as follows:

$$\begin{cases} q_\psi(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0) = \mathcal{N}\left(\tilde{\boldsymbol{\mu}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0), \tilde{\boldsymbol{\Sigma}}_t \mathbf{I}\right), \quad \text{where } \tilde{\boldsymbol{\Sigma}}_t = \dfrac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}, \\ \tilde{\boldsymbol{\mu}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0) = \dfrac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}(\boldsymbol{\theta}_t - \mathbf{K}_t) + \mathbf{K}_{t-1} + \dfrac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{\theta}_0, \end{cases} \tag{6}$$

in which $\kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t)$ is abbreviated as $\mathbf{K}_t$. We place its proof in Appendix B.3 (Theorem 1).

4.2 Algorithmic Implementation of KDDM

Similar to prior works (Yuan & Qiao, 2024; Li & He, 2026), we parameterize **KDDM** using a trainable *reconstructor* $\hat{\boldsymbol{\theta}}_0$ that aims to recover a clean trajectory chunk $\boldsymbol{\theta}_0$ from $\boldsymbol{\theta}_t$ at every diffusion timestep t. The underlying model can use any common NN architectures, such as 1D temporal CNN (Janner et al., 2022) and Time-Series Diffusion Transformer (Chi et al., 2025).

Training of **KDDM** is performed by optimizing the usual variational evidence lower bound (ELBO):

$$\mathbb{E}_{q_{\text{data}}(\boldsymbol{\theta}_0)}[\log p(\boldsymbol{\theta}_0)] \geq -\mathbb{E}_{q_\psi(\boldsymbol{\theta}_{0:T})}\left[\log \frac{q_\psi(\boldsymbol{\theta}_{1:T}|\boldsymbol{\theta}_0)}{p(\boldsymbol{\theta}_{0:T})}\right] := \text{ELBO} \tag{7}$$

In the following inequality, we provide a closed-form expression of the ELBO on the maximum likelihood objective; please refer to Theorem 2 in Appendix B.4 for derivations.

$$-\text{ELBO} \leq \mathbf{C} + \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\theta}_0, \boldsymbol{\epsilon}}\left[k_t \left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t) + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, t\right)\right\|_2^2\right] \tag{8}$$

where $k_{1:T}$ are a set of positive coefficients, and $\mathbf{C}$ is the constant that is irrelevant for optimization purpose. In our implementation, it is reduced to minimizing the unweighted objective:

$$\mathcal{L}_{\text{simple}} := \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{\theta}_0, \boldsymbol{\theta}_t) \sim q_{\text{data}}(\boldsymbol{\theta}_0) q_\psi(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0)}\left[\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0(\boldsymbol{\theta}_t, t)\right\|_2^2\right]. \tag{9}$$

Once training is completed, the sampling of **KDDM** proceeds by alternating the mean update steps, as shown in Eqn. (6), and the addition of Gaussian noise with variance σ_t. We summarize the training process in Alg. 1, and the sampling process in Alg. 2, both of which are straightforward to implement.

4.3 Generalized and Enhanced Version

KDDM can generate high-quality robotic trajectories but lacks flexibility regarding the computational time and diversity under the condition $\mathbf{c}$. To overcome these, we make the following three improvements to **KDDM**, enabling it to be readily incorporated into diffusion-based systems today.

① **Generalized sampling with ODEs.** Fast inference is critical for closed-loop real-time control; hence, ODE-based sampling methods such as *DDIM* (Song et al., 2020a; Zhang et al., 2023b) play an important role in existing diffusion policies. To enable similar acceleration, we extend **KDDM** to a non-Markovian process with equivalent training objective and marginals, dubbed **KDIM**, whose derivation is provided in Appendix B.5.1 (Proposition 1). One can then generate $\boldsymbol{\theta}_{t-\Delta t}$ from $\boldsymbol{\theta}_t$ via

$$\boldsymbol{\theta}_{t-\Delta t} = \sqrt{\bar{\alpha}_{t-\Delta t}}\,\hat{\boldsymbol{\theta}}_0 + \hat{\mathbf{K}}_{t-\Delta t} + \sqrt{1-\bar{\alpha}_{t-\Delta t}-\sigma_t^2} \cdot \frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\,\hat{\boldsymbol{\theta}}_0 - \hat{\mathbf{K}}_t}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t \boldsymbol{\epsilon}_t \tag{10}$$

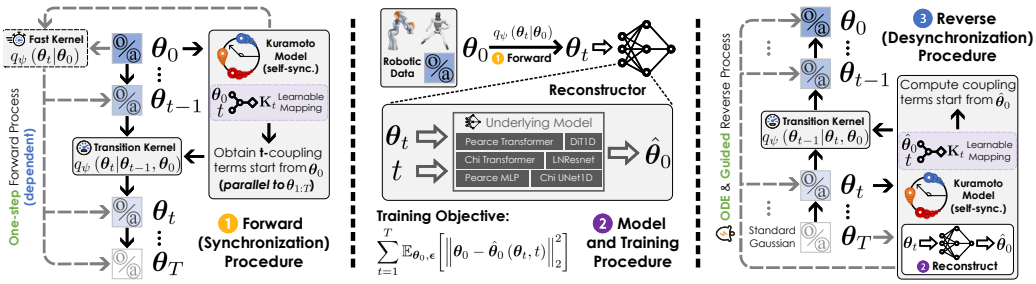


Figure 6: **Overview of our KDDM.** Combined with the Kuramoto model, it elegantly and efficiently incorporates spatio-temporal dependencies within trajectories into diffusion-based decision making.

where Δt denotes the number of steps to skip, and $\hat{\mathbf{K}}_t$ is the abbreviation of $\kappa_t \mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t\right)$. Consider the special case when $\sigma_t = 0$ for all t, KDIM deterministically maps noise back to the data samples.

② **Conditioned desynchronization processes.** We may also consider guiding the reverse process toward trajectories that satisfy specific conditions or exhibit desired behaviors, i.e., approximating the conditional distribution $p(\boldsymbol{\theta}_0|\mathbf{c})$. In addition to employing well designed architectures, e.g., visual encoders for raw image sequence (Black et al., 2024), we seek to condition the desynchronization process of KDDM with extra adaptive-temperature guidance (Ajay et al., 2023). Here, we can steer its trajectory through the following three classes of Plug-and-Play (PnP) strategies, which covers a wide range of prevalent algorithms (Dhariwal & Nichol, 2021; Ye et al., 2024; Lai et al., 2025); due to space limits, we defer their details (e.g., algorithms and theories) to Appendix B.5.2.

• Training-Based Guidance: One approach could be to train a time-conditional classifier $p_\phi(\mathbf{c}|\boldsymbol{\theta}_t)$ on noisy trajectories $\boldsymbol{\theta}_t$, known as *Classifier Guidance* (CG). By using the Bayesian decomposition, its gradient provides the guidance term: $\nabla_{\boldsymbol{\theta}_t} \log p_\phi(\mathbf{c}|\boldsymbol{\theta}_t) \approx \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\mathbf{c}|\boldsymbol{\theta}_t)$. Further, KDDM also applies to *Classifier-Free Guidance* (CFG), where the unconditional model is jointly learned.

• Training-Free Guidance: Based on the integral $q_\psi(\mathbf{c}|\boldsymbol{\theta}_t) = \int q(\mathbf{c}|\boldsymbol{\theta}_0)\, q_\psi(\boldsymbol{\theta}_0|\boldsymbol{\theta}_t)\, d\boldsymbol{\theta}_0$, training-free guidance avoids calculating $q_\psi(\mathbf{c}|\boldsymbol{\theta}_t)$ directly. Instead, it leverages a $\mathbf{c}$-related metric, e.g., the reward, and defines a surrogate term $\nabla_{\boldsymbol{\theta}_t} \textsc{Metrics}\left(\boldsymbol{\theta}_{0|t}, \mathbf{c}\right) \approx \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\mathbf{c}|\boldsymbol{\theta}_t)$. We propose two different methods for KDDM in such a training-free manner — *Replacement* for inpainting tasks (Janner et al., 2022; Li et al., 2024a) and *Posterior Sampling* for border tasks (Lee & Choi, 2025; Hu et al., 2026).

• Sequential Monte Carlo: The third strategy, also known as particle filtering, has the good property that, as the number of particles $\to \infty$, the sampler converges to the true posterior distribution (Chopin et al., 2020). In KDDM, the particles or candidates, are propagated through the reverse sampling steps, re-weighted according to the metric above. It is gradient-free and sampling-only (Lu et al., 2025b).

③ **Learning to synchronize quickly.** As mentioned before in §4.1, it is possible to learn the coupling terms $\{\mathbf{K}_\psi(\boldsymbol{\theta}_0, t)\}_{t=1}^T$. Therefore, we can employ a time-conditional dependency network (e.g., MLPs) that directly predict $\mathbf{K}_t$ given a pair of inputs $(\boldsymbol{\theta}_0, t)$. The time complexity of each self-synchronization is thereby reduced from $\mathcal{O}(t)$ to $\mathcal{O}(1)$, with just a supervised pre-training procedure required. Despite the potential for overfitting, we find it generally performs well in our evaluations.

4.4 Putting Everything Together

End-to-End View. Fig. 6 gives an overview of KDDM, which consists of three procedures (from left to right): synchronization (forward), training, and desynchronization (reverse). Our framework often commences with pretraining a dependency mapping $(\boldsymbol{\theta}_0, t) \to \mathbf{K}_t$. The forward procedure then generates training data, i.e., tuples $(t, \boldsymbol{\theta}_0, \boldsymbol{\theta}_t)$, in a single step according to Eqn. (5). The training procedure receives them and learns a model-agnostic reconstructor $(\boldsymbol{\theta}_t, t) \to \hat{\boldsymbol{\theta}}_0$ using the objective in Eqn. (9). When it comes to decision-making time, the reverse procedure generates the required trajectories using the SDE in Eqn. (6), the ODE in Eqn. (10), or controllable sampling in §4.3.

Benefits. From a comparative perspective, our proposed method has several desirable properties. First, compared with classical diffusion, KDDM incorporates spatio-temporal dependencies into both the forward and reverse processes in an explicit manner. Second, compared with kuramoto diffusion, KDDM overcomes the shackles of Kuramoto's non-linearity, making it more compatible with the decision-making ecosystem. Further, we can derive that KDDM has an upper bound of negative log likelihood than others (see Appendix B.6).

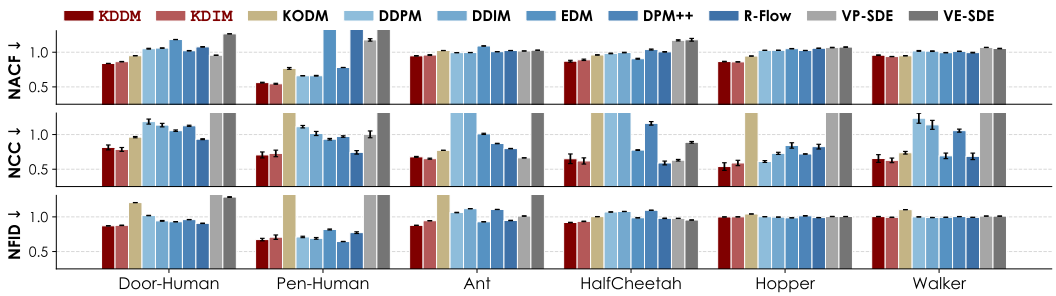


Figure 7: Data quality comparison with representative DMs on Adroit Hand (Door-Human and Pen-Human) and MuJoCo (Ant, HalfCheetah, Hopper and Walker). The results are median-normalized.

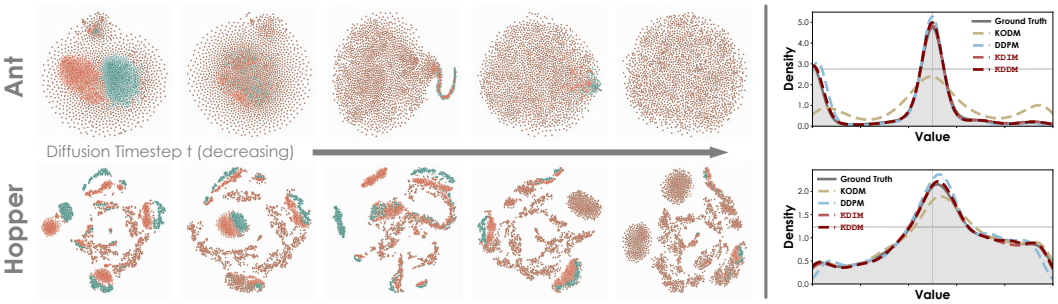


Figure 8: **Left:** Visualization of the coupling terms $\mathbf{K}_\psi(\boldsymbol{\theta}_0, t)$ in the reverse process of **KDDM** and the GT forward process using t-SNE. **Right:** Comparing the learned distributions of DMs using KDE.

5 Evaluation

We extensively evaluate the merits of **KDDM** as a generative backbone model across diverse decision-making applications in trajectory augmentation, stitching, planning (RL), and robotic control (IL). Please find the dataset, baseline, metric, and other reproducibility details in Appendix C. We also conduct additional experiments (e.g., omitted results and ablations) in Appendix D, and include visualizations in Appendix E. Our evaluation in this section studies the following research questions:

Q1 Does and how **KDDM** produce smoother and more realistic trajectories than other DMs?

Q2 Does introducing the Kuramoto model make a better diffusion-based planner in offline RL?

Q3 Does **KDDM** learn effective (multi-modal) robotic control from human demonstrations?

Q4 Does **KDDM** enable real-world robot deployment on diffusion policies?

5.1 Data Synthesizer for Trajectory Augmentation

We first highlight the advantages of **KDDM** for unconditional trajectory generation (Q1). Specifically, we compare the proposed method against **8** state-of-the-art (SOTA) DMs in the MuJoCo and Adroit Hand environments (Parmas et al., 2018; Fu et al., 2020), using a vanilla Transformer as the underlying model architecture. The collected trajectories are chunked into sequences of length 100, and we evaluate the quality of the generated data (with $\sim$100 diffusion timesteps) along three dimensions: temporal (*NACF*), spatial (*NCC*), and distributional (*NFID*) fidelity. As shown in Fig. 7, our proposal clearly outperforms the other DMs in spatio-temporal fidelity and ranks among the top performers in distributional fidelity. Quantitatively, compared with the best-performing baseline, **KDDM** improves NACF and NCC by up to 15.6% and 12.7%, respectively. Although KODM exhibits unstable spatial correlations, its NACF ranks, overall, behind only our methods, suggesting the beneficial effect of synchronization. We attribute such instability to its limited capability in distributional modeling, as evidenced by its average NFID being 1.95$\times$ that of **KDDM**. Also note that **KDIM** and **KDDM** are closely matched in performance, which is consistent with the expected behavior of the ODE formulation.

For a better understanding, we further apply t-SNE (Van der Maaten & Hinton, 2008) to the coupling terms from **KDDM**, i.e., $\mathbf{K}_\psi(\hat{\boldsymbol{\theta}}_0, t)$, and ground-truth (GT) data $\mathbf{K}_\psi(\boldsymbol{\theta}_0, t)$ (Fig. 8, left), in which greater overlap indicates higher similarity. Except for large t, where >90% of terms fall within $[-10^{-2}, 10^{-2}]$, the two distributions match well in general, meaning that **KDDM** effectively captures spatio-temporal dependencies for smoothness. Moreover, our Kernel Density Estimation (KDE)

Table 1: **Evaluation Results of Offline RL Benchmark.** The normalized scores of diffusion-based planners on the D4RL benchmark. Results correspond to the mean and standard error over 150 episode seeds. Part of the results are obtained from Ki et al. (2026). Best performance in **bold**, second-best underlined.

Dataset	Environment	BC	Diffuser	DD	AD	HD	Dlite	DV	PG	KDDM	KDIM
Medium-Expert	HalfCheetah	35.8	88.9±0.3	90.6±1.3	89.6±0.8	92.5±0.3	90.8±0.9	92.3±0.6	**95.2**±0.1	94.1±1.2	93.3±2.0
	Hopper	111.9	103.3±1.3	111.8±1.8	111.6±2.0	**115.3**±1.1	110.3±0.3	109.9±1.0	110.4±0.1	110.9±0.4	111.6±1.3
	Walker2d	6.4	106.9±0.2	108.8±1.7	108.2±0.8	107.1±1.1	106.4±0.3	109.0±0.1	109.4±0.1	**110.2**±0.3	109.5±0.5
Medium	HalfCheetah	36.1	42.8±0.3	49.1±1.0	44.2±0.6	46.7±0.2	48.6±0.7	50.9±0.1	45.6±0.5	50.7±0.7	**51.1**±0.2
	Hopper	29.0	74.3±1.4	79.3±3.6	96.6±2.7	99.3±0.3	**99.5**±0.7	84.1±2.5	97.5±0.6	96.4±1.5	96.8±2.7
	Walker2d	6.6	79.6±0.6	82.5±1.4	84.4±2.6	84.0±0.6	**85.1**±0.5	79.5±0.3	82.3±0.2	84.5±0.8	82.8±2.2
Medium-Replay	HalfCheetah	38.4	37.7±0.0	39.3±4.1	38.3±0.9	38.1±0.7	42.9±0.4	46.4±0.2	46.4±0.4	**46.8**±0.7	46.5±0.6
	Hopper	11.3	93.6±0.4	**100.0**±0.7	92.2±1.5	94.7±0.7	97.8±1.3	91.3±0.1	91.3±0.3	94.7±0.5	93.0±1.1
	Walker2d	11.8	70.6±1.6	75.0±4.3	**84.7**±3.1	84.1±2.2	84.6±1.7	83.5±2.5	83.7±1.0	84.0±2.6	84.1±1.4
Average		31.9	77.5	83.3	81.8	84.6	85.1	83.0	84.6	**85.8**	85.4
Mixed	Kitchen	47.5	52.5±2.5	65.0±2.8	51.8±0.8	71.7±2.7	71.9±1.4	73.3±0.6	**74.6**±0.4	73.8±0.2	72.7±1.3
Partial	Kitchen	33.8	55.7±1.3	57.0±2.5	55.5±0.4	73.3±1.4	69.9±0.7	76.2±4.3	88.0±3.1	**93.2**±1.6	91.4±2.8
Average		40.7	54.1	61.0	53.7	72.5	70.9	74.8	81.3	**83.5**	82.1
Play	Antmaze-Medium	0.0	6.7±5.7	8.0±4.6	12.0±7.5	–	88.0±2.2	80.3±1.3	87.8±1.5	**88.7**±1.5	87.4±1.6
	Antmaze-Large	0.0	17.3±1.9	0.0±0.0	5.3±3.4	–	72.4±2.3	80.8±2.8	**82.4**±2.1	76.0±2.4	78.2±2.0
Diverse	Antmaze-Medium	0.0	2.0±1.6	4.0±2.8	6.0±3.3	88.7±8.1	**89.2**±2.0	82.0±3.6	87.1±1.9	87.4±1.6	85.6±1.7
	Antmaze-Large	0.0	27.3±2.4	0.0±0.0	8.7±2.5	**83.6**±5.8	80.4±5.1	76.0±2.3	76.0±1.9	80.2±2.1	78.6±2.8
Average		0.0	13.3	3.0	8.0	–	82.5	79.9	**83.4**	83.0	82.4
Maze2D	U-Maze	3.8	113.9±3.1	–	135.1±5.8	128.4±3.6	–	133.8±1.9	139.2±1.3	**142.2**±2.4	140.0±4.5
	Medium	30.5	121.5±2.7	–	129.9±4.6	135.6±3.0	–	144.1±2.1	**159.5**±0.8	150.8±1.0	151.4±4.2
	Large	5.0	123.0±6.4	–	167.9±5.0	155.8±2.5	–	197.6±4.2	195.2±2.3	**202.9**±1.3	198.8±5.3
Average		13.0	119.5	–	144.3	139.9	–	158.5	164.6	**165.3**	163.4

shows this does not come at the expense of the realness (Fig. 8, right): compared with DDPM, our method better matches the GT curve, while KODM performs much worse; these are in line with our theoretical results (Appendix B.6). We omit the results for specific conditional generation tasks, i.e., trajectory stitching or inpainting; however, it can be found in Appendix D.1.

5.2 Planner for Offline Reinforcement Learning

We next demonstrate that our methods can improve offline reinforcement learning with long and short horizons (Q2). We conducted experiments on the D4RL dataset (Fu et al., 2020), one of the most widely used benchmarks for offline RL, covering robot locomotion, robot manipulation, and maze navigation. We compare to **7** prior algorithms panning other DMs to data-driven planning implemented by `CleanDiffuser` (Dong et al., 2024b), including *Diffuser* (Janner et al., 2022), Decision Diffuser (*DD*, Ajay et al., 2023), AdaptDiffuser (*AD*, Liang et al., 2023), Hierarchical Diffuser (*HD*, Chen et al., 2024b), DiffuserLite (*Dlite*, Dong et al., 2024a), Diffusion Veteran (*DV*, Lu et al., 2025b) and Prior Guidance (*PG*, Ki et al., 2026). Here apart from the used diffusion backbone, **KDDM** and **KDIM** are built on top of DV, following its design choices (e.g., Monte Carlo and inverse dynamics). Quantitative results are summarized in Table 1.

As can be seen in the table, our method achieves the overall best average performance, outperforming previous diffusion-based planners while surpassing the baseline DV. Notably, **KDDM** improves the average D4RL normalized score by 2.4 points on Gym-MuJoCo, 7.3 points on Kitchen, 3.5 points on AntMaze, and 2.5 points on Maze2D, compared with DV. In addition, across all tasks and the mean (totaling 22 results), **KDDM** and **KDIM** lead all planners with 13 and 9 (near-)optimal results respectively. We attribute these gains largely to the correction or establishment of spatio-temporal dependencies during high-level planning by our methods.

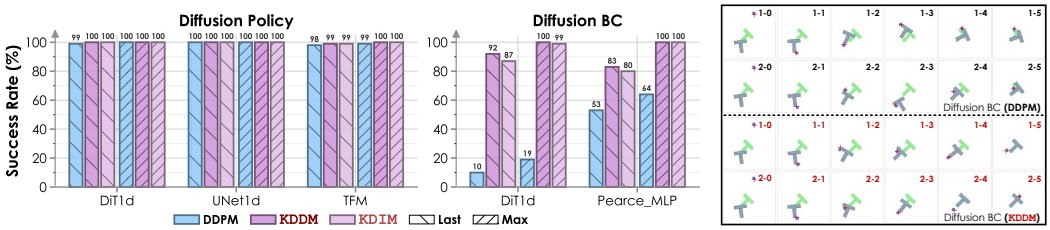
5.3 Policy for Offline Imitation Learning

In this subsection, we investigate the offline imitation learning setting to assess how effectively our methods support generative control policies (Q3). We evaluate the popular *Diffusion Policy* (Chi et al., 2025), *Diffusion BC* (Pearce et al., 2023), and *Diffusion Steering via Reinforcement Learning* (*SRL*, Wagenmaker et al., 2025) with different diffusion backbones on the Robomimic (Mandlekar et al., 2021), PushT (Florence et al., 2022), and Relay-Kitchen (Gupta et al., 2019) benchmarks. We also include the RNN-based LSTM-GMM (Mandlekar et al., 2021) and Transformer-based ACT (Zhao et al., 2023) as representative non-diffusion baselines for comparison. Each method is evaluated in its best-performing action space, with position control used for Diffusion Policy, Diffusion SRL, and ACT, and velocity control for the others.

Table 2 summarizes our results for 12 state-based tasks. We see that the improvements brought by our proposed DM are pervasive (approximately 90% of the best-performing results), demonstrating

Table 2: **Evaluation Results of Offline IL Benchmark.** The metrics show success rate for Robomimic and Relay-Kitchen, target area coverage for PushT. We report mean performance of last checkpoint denoted as LAST and max performance of the last 10 checkpoints denoted as MAX, with each averaged over 3 seeds and 50 episodes. We show the performance of (LAST / MAX). For each diffusion-based control policy, best results are shown in strong positive colors, comparable ones in mild positives.

Task Name	LSTM-GMM*	ACT	Diffusion Policy			Diffusion BC			Diffusion SRL		
			DDPM	KDDM	KDIM	DDPM	KDDM	KDIM	DDPM	KDDM	KDIM
pusht	0.59/0.70	0.99/**1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	0.68/0.99	0.83/**1.00**	0.80/0.99	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**
pusht-keypoints	0.61/0.67	0.99/**1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	0.99/0.99	0.99/**1.00**	0.99/**1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**
relay-kitchen	0.75/0.79	0.72/0.76	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	0.68/0.80	0.71/0.85	0.74/0.84	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**
lift-ph	0.96/**1.00**	0.98/**1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	0.97/**1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**
lift-mh	0.93/**1.00**	0.98/**1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	0.92/**1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**
can-ph	0.91/**1.00**	0.92/0.98	0.99/**1.00**	0.99/**1.00**	0.99/**1.00**	0.95/0.99	0.99/**1.00**	0.99/**1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**
can-mh	0.81/**1.00**	0.90/0.98	0.97/0.99	**0.99**/0.99	0.98/0.99	0.67/0.91	0.78/0.94	0.74/0.88	0.92/0.96	**0.99**/0.99	0.95/0.99
square-ph	0.73/0.95	0.80/0.90	0.59/0.97	0.68/**0.99**	0.66/0.95	0.66/0.80	0.54/0.80	0.55/0.77	0.72/0.96	**0.83/0.99**	0.80/0.94
square-mh	0.59/0.86	0.46/0.72	0.87/**0.96**	0.79/0.93	0.75/0.88	0.42/0.52	0.45/0.60	0.42/0.50	**0.83**/0.84	0.75/0.94	0.72/0.89
transport-ph	0.47/0.76	0.64/0.85	0.78/0.83	0.81/0.88	**0.82**/0.82	0.11/0.20	0.16/0.34	0.14/0.32	0.74/0.82	0.75/**0.92**	0.75/0.90
transport-mh	0.20/0.62	0.40/0.68	**0.58**/0.72	0.54/**0.75**	0.51/0.68	0.00/0.04	0.02/0.10	0.01/0.06	0.42/0.67	0.49/0.72	0.46/0.68
toolhang-ph	0.31/0.67	0.64/0.82	0.79/0.90	0.83/**0.92**	**0.84**/0.90	0.19/0.32	0.20/0.44	0.18/0.40	0.38/0.70	0.45/0.70	0.45/0.66
Average	0.66/0.84	0.79/0.89	0.88/0.94	**0.89/0.96**	**0.89**/0.94	0.60/0.71	0.64/0.76	0.63/0.73	0.85/0.93	0.86/0.94	0.85/0.93



(a) Success rates with different network architectures (b) Visualized cloned behaviors

Figure 9: Performance comparison and interaction examples of image-based tasks on PushT-Image.

its effectiveness at capturing dependencies from human data. We highlight in particular that **KDDM** and **KDIM** substantially improve over DDPM-based BC, with success rates increasing by more than 10% sometimes. To evaluate our method on image-based tasks, we also evaluate on PushT-Image with different NN for visual information conditioning in Fig. 9. As these figures illustrate, we find **KDDM**: (i) achieves near-perfect success rates in most cases; (ii) is indeed affected by the architecture; (iii) greatly surpasses DDPM-BC in both quantitative performance (up to 5+ times success rate) and qualitative demonstrations as presented in Fig. 9(b). In consequence, **KDDM** serves as a PnP component that boosts diffusion-based robotic control performance.

5.4 REAL-WORLD ROBOT MANIPULATION



Figure 10: Four tasks.

Finally, we demonstrate that **KDDM** enables effective adaptation of diffusion policies in the real world (Q4). We evaluate on a SO-ARM100/SO-ARM101 6-DoF robot arm using the `lerobot` (Cadene et al., 2026) setup and manually collected datasets. As shown in Fig. 10, the following four different tasks are considered: `put-the-knife`, `put-the-egg`, `pick-the-bottle`, and `place-the-duck`. Please refer to our website and Appendix E.4 for visualizations. In the experiments, the success rate ranges from 5/10 to 8/10, compared with 2/10 for diffusion policy in the literature (Wagenmaker et al., 2025).

6 CONCLUSION

In this paper, we aim to improve DMs in decision-making scenarios, motivated by 1) the neglect of spatio-temporal dependencies in prior works, and 2) the lack of a practical solution for kuramoto diffusion processes. We propose **KDDM**, a physics-inspired approach that encodes chunk-level spatio-temporal patterns of all trajectories into forward and reverse procedures through (de)synchronization. The new diffusion paradigm combines ease of training and implementation with theoretical soundness. We then generalize **KDDM** to **KDIM**, the accelerated DDIM-style ODE version, and then equip them with conditioned strategies. Experiments verify **KDDM**'s effectiveness and feasibility for coherent decision making with negligible compatibility overhead in both simulation and real-world settings. Due to space constraints here, we defer discussion of several *limitations* regarding **KDDM** and potential *future directions* to Appendix F and Appendix G respectively.

7 AI USE STATEMENT

Generative AI tools, including CHATGPT, were used in this work to assist with language polishing, grammatical refinement, and code debugging. All AI-assisted content was carefully reviewed, revised, and validated by the authors. The authors assume full responsibility for the accuracy, originality, and integrity of the manuscript.

8 ETHICS STATEMENT

In this paper, we do not use any non-public data, unauthorized software, or unauthorized APIs. The real-world robotic data used in our experiments were collected by the authors using our own robotic hardware and do not involve human subjects or personally identifiable information. Therefore, we do not identify any privacy or other ethical concerns arising from the data or software used in this work.

9 REPRODUCIBILITY STATEMENT

The paper provides all information necessary to reproduce the main experimental results insofar as they support the paper's primary claims and conclusions. To be more specific, we provide detailed descriptions of the experimental setups and implementation details in Appendix C. In addition, our experimental codebase is included in the ***supplementary material*** to facilitate reproduction.

REFERENCES

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. Cited on page 40.

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985. Cited on page 20.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023. Cited on pages 3, 6, 8, 20, 38, and 45.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5096–5103. IEEE, 2024. Cited on page 21.

Haldun Balim, Na Li, and Yilun Du. Model-based diffusion sampling for predictive control in offline decision making. *arXiv preprint arXiv:2512.08280*, 2025. Cited on page 20.

David Bell, Yujie Lu, Shinda Huang, William Wang, and Amy Zhang. Language control diffusion: Efficiently scaling through space, time, and tasks. In *International Conference on Learning Representations*, volume 2024, pp. 51378–51408, 2024. Cited on page 21.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. Cited on page 22.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. Cited on page 21.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. Cited on pages 1, 6, and 21.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. Cited on page 36.

Remi Cadene, Simon Alibert, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, et al. Lerobot: An open-source library for end-to-end robot learning. In *International Conference on Learning Representations*, volume 2026, pp. 122398–122417, 2026. Cited on pages 9, 21, and 41.

Gabriel Cardoso, Sylvain Le Corff, Eric Moulines, et al. Monte carlo guided denoising diffusion models for bayesian linear inverse problems. In *International Conference on Learning Representations*, volume 2024, pp. 44001–44037, 2024. Cited on pages 20 and 32.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. *arXiv preprint arXiv:2502.02316*, 2025. Cited on page 21.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024a. Cited on page 3.

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. In *The Twelfth International Conference on Learning Representations*, 2024b. Cited on pages 8 and 20.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022. Cited on pages 21 and 45.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. Cited on page 3.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. Cited on pages 1, 3, 5, 8, 21, 39, and 42.

Daesol Cho, Dongseok Shim, and H Jin Kim. S2p: State-conditioned image synthesis for data augmentation in offline reinforcement learning. *Advances in neural information processing systems*, 35:11534–11546, 2022. Cited on page 21.

Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020. Cited on pages 6, 21, and 22.

Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in neural information processing systems*, 35:25683–25696, 2022. Cited on page 20.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. Cited on pages 20 and 30.

Jiawen Dai and Yue Song. Winfree oscillatory neural network. *arXiv preprint arXiv:2605.20922*, 2026. Cited on page 52.

Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions. *arXiv preprint arXiv:2209.05442*, 2022. Cited on page 20.

Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15617–15625. IEEE, 2025. Cited on page 21.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. Cited on pages 6, 20, and 29.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024. Cited on page 21.

Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. Maximum entropy reinforcement learning with diffusion policy. *arXiv preprint arXiv:2502.11612*, 2025. Cited on page 21.

Zibin Dong, Jianye Hao, Yifu Yuan, Fei Ni, Yitian Wang, Pengyi Li, and Yan Zheng. Diffuserlite: Towards real-time diffusion planning. *Advances in Neural Information Processing Systems*, 37: 122556–122583, 2024a. Cited on pages 8, 20, and 39.

Zibin Dong, Yifu Yuan, Jianye Hao, Fei Ni, Yi Ma, Pengyi Li, and Yan Zheng. Cleandiffuser: An easy-to-use modularized library for diffusion models in decision making. *Advances in Neural Information Processing Systems*, 37:86899–86926, 2024b. Cited on pages 3, 8, 20, 43, and 45.

G Bard Ermentrout. Synchronization in a pool of mutually coupled oscillators with random frequencies. *Journal of Mathematical Biology*, 22(1):1–9, 1985. Cited on page 2.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pp. 158–168. PMLR, 2022. Cited on pages 8 and 37.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. Cited on pages 7, 8, 36, and 37.

Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994. Cited on page 3.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019. Cited on pages 8 and 37.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. Cited on page 21.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023. Cited on page 21.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, WeiHsiang Liao, Yuki Mitsufuji, Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. In *International Conference on Learning Representations*, volume 2024, pp. 44819–44850, 2024. Cited on page 20.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2022. Cited on pages 20 and 29.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. Cited on pages 1, 3, 20, 21, 38, and 41.

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. Cited on pages 20 and 30.

Emiel Hoogeboom and Tim Salimans. Blurring diffusion models. *arXiv preprint arXiv:2209.05557*, 2022. Cited on page 20.

Jifeng Hu, Yanchao Sun, Sili Huang, SiYuan Guo, Hechang Chen, Li Shen, Lichao Sun, Yi Chang, and Dacheng Tao. Instructed diffuser with temporal condition guidance for offline reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. Cited on page 6.

Chin-Wei Huang, Jae Hyun Lim, and Aaron C Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34: 22863–22876, 2021. Cited on page 20.

Baris Imre. An investigation of generative replay in deep reinforcement learning. In *34th Twente Student Conference on IT*, 2021. Cited on page 21.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a. Cited on page 1.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025b. Cited on pages 1 and 21.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. Cited on pages 1 and 21.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022. Cited on pages 1, 3, 5, 6, 8, 20, 38, and 47.

Adam Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. 2009. Cited on page 32.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 67195–67212, 2023. Cited on page 45.

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. Cited on pages 3, 20, and 38.

Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. *Advances in neural information processing systems*, 34:21757–21769, 2021. Cited on page 20.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in neural information processing systems*, 35:23593–23606, 2022. Cited on page 20.

Donghyeon Ki, JunHyeok Oh, Seong-Woong Shim, and Byung-Jun Lee. Prior-guided diffusion planning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 38:59790–59820, 2026. Cited on pages 8, 20, and 39.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Cited on page 41.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. Cited on pages 20 and 24.

Yoshiki Kuramoto. Chemical turbulence. In *Chemical oscillations, waves, and turbulence*, pp. 111–140. Springer, 1984. Cited on pages 1 and 2.

Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models. *arXiv preprint arXiv:2510.21890*, 2025. Cited on pages 1, 6, 20, 21, and 52.

Kyowoon Lee and Jaesik Choi. Local manifold approximation and projection for manifold-aware diffusion planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 32987–33004. PMLR, 2025. Cited on page 6.

Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. Diffstitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 28597–28609. PMLR, 2024a. Cited on pages 6, 21, and 42.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026. Cited on page 5.

Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pp. 20035–20064. PMLR, 2023. Cited on pages 20 and 39.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022. Cited on pages 20 and 30.

Zechu Li, Rickmer Krohn, Tao Chen, Anurag Ajay, Pulkit Agrawal, and Georgia Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *Advances in Neural Information Processing Systems*, 37:38456–38479, 2024b. Cited on page 21.

Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: diffusion models as adaptive self-evolving planners. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 20725–20745, 2023. Cited on pages 8, 20, and 39.

Shujian Liao, Hao Ni, Marc Sabate-Vidales, Lukasz Szpruch, Magnus Wiese, and Baoren Xiao. Sig-wasserstein gans for conditional time series generation. *Mathematical Finance*, 34(2):622–670, 2024. Cited on page 39.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. Cited on pages 20 and 38.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024. Cited on page 20.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. Cited on pages 1, 3, and 20.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022. Cited on pages 3, 20, and 38.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025a. Cited on pages 3, 20, and 38.

Cong Lu, Philip Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. *Advances in Neural Information Processing Systems*, 36:46323–46344, 2023. Cited on pages 3 and 21.

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *The Thirteenth International Conference on Learning Representations*, 2025b. Cited on pages 3, 6, 8, 20, 32, and 39.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11461–11471, 2022. Cited on pages 20 and 30.

Yunhao Luo, Utkarsh Mishra, Yilun Du, and Danfei Xu. Generative trajectory stitching through diffusion composition. *Advances in Neural Information Processing Systems*, 38:37809–37843, 2026. Cited on page 42.

Lei Lyu, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 93967–93990, 2026. Cited on page 21.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021. Cited on pages 8 and 37.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PmLR, 2021. Cited on pages 20, 38, and 41.

Giorgio Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981. Cited on page 3.

Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. Pipps: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, pp. 4065–4074. PMLR, 2018. Cited on page 7.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. Cited on pages 3, 8, 21, and 39.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in neural information processing systems*, 34:12208–12221, 2021. Cited on page 36.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023. Cited on page 21.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017. Cited on page 36.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015. Cited on page 20.

Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. *arXiv preprint arXiv:2206.13397*, 2022. Cited on pages 20 and 52.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. Cited on page 21.

Adrian Smith. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013. Cited on page 32.

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. Cited on pages 3, 20, and 21.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a. Cited on pages 3, 5, 20, and 38.

Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pp. 32483–32498. PMLR, 2023. Cited on pages 20 and 30.

Mingchen Song, Xiang Deng, Zhiling Zhou, Jie Wei, Weili Guan, and Liqiang Nie. A survey on diffusion policy for robotic manipulation: Taxonomy, analysis, and future directions. 2025. Cited on pages 1 and 20.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. Cited on pages 20, 21, and 28.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b. Cited on pages 20, 27, and 38.

Yue Song, Andy Keller, Sevan Brodjian, Takeru Miyato, Yisong Yue, Pietro Perona, and Max Welling. Kuramoto orientation diffusion models. *Advances in Neural Information Processing Systems*, 38: 4047–4076, 2026. Cited on pages 1, 3, 20, 21, 28, 34, 45, and 52.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 63–70. IEEE, 2024. Cited on page 21.

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913. IEEE, 2012. Cited on page 2.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. Cited on page 21.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023. Cited on page 20.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun Kg, Rodrigo Perez-Vicente, et al. Gymnasium: A standard interface for reinforcement learning environments. *Advances in Neural Information Processing Systems*, 38, 2026. Cited on page 36.

Brian L Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi S Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023. Cited on pages 20 and 32.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. Cited on page 7.

JL Van Hemmen and WF Wreszinski. Lyapunov function for the kuramoto model of nonlinearly coupled oscillators. *Journal of Statistical Physics*, 72(1):145–166, 1993. Cited on page 2.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. Cited on page 3.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025. Cited on pages 8, 9, 21, and 39.

Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *arXiv preprint arXiv:2212.00490*, 2022a. Cited on page 20.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022b. Cited on page 21.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011. Cited on page 28.

Arthur T Winfree. Biological rhythms and the behavior of populations of coupled oscillators. *Journal of theoretical biology*, 16(1):15–42, 1967. Cited on page 2.

Rosa Wolf, Yitian Shi, Sheng Liu, and Rania Rayyes. Diffusion models for robotic manipulation: A survey. *Frontiers in Robotics and AI*, 12:1606247, 2025. Cited on pages 1 and 20.

Wei Xiao, Johnson Wang, Chuang Gan, Ramin Hasani, Mathias Lechner, and Daniela Rus. Safediffuser: Safe planning with diffusion probabilistic models. In *International Conference on Learning Representations*, volume 2025, pp. 100391–100412, 2025. Cited on page 20.

Cheng-Fu Yang, Haoyang Xu, Te-Lin Wu, Xiaofeng Gao, Kai-Wei Chang, and Feng Gao. Planning as in-painting: A diffusion-based embodied task planning framework for environments under uncertainty. *arXiv preprint arXiv:2312.01097*, 2023. Cited on page 20.

Ling Yang, Zhilong Zhang, Zhaochen Yu, Jingwei Liu, Minkai Xu, Stefano Ermon, and Bin Cui. Cross-modal contextualized diffusion models for text-guided visual generation and editing. In *International Conference on Learning Representations*, volume 2024, pp. 13972–13997, 2024. Cited on pages 20 and 21.

Qianlan Yang and Yu-Xiong Wang. Rtdiff: Reverse trajectory synthesis via diffusion for offline reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 28194–28212, 2025. Cited on page 21.

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. Cited on pages 1, 6, 20, and 29.

Jaesik Yoon, Hyeonseo Cho, Doojin Baek, Yoshua Bengio, and Sungjin Ahn. Monte carlo tree diffusion for system 2 planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 72618–72640. PMLR, 2025. Cited on page 3.

Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. Minari, September 2024. URL https://doi.org/10.5281/zenodo.13767625. Cited on page 36.

Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. In *International Conference on Learning Representations*, volume 2024, pp. 41582–41610, 2024. Cited on pages 5 and 30.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024. Cited on page 21.

Jichen Zhang, Liqun Zhao, Antonis Papachristodoulou, and Jack Umenberger. Constrained diffusers for safe planning and control. *Advances in Neural Information Processing Systems*, 38:34965–34998, 2026a. Cited on page 20.

Junjie Zhang, Jiafei Lyu, Xiaoteng Ma, Jiangpeng Yan, Jun Yang, Le Wan, and Xiu Li. Uncertainty-driven trajectory truncation for data augmentation in offline reinforcement learning. In *ECAI 2023: 26th European Conference on Artificial Intelligence, September 30–October 4, 2023, Kraków, Poland–Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, pp. 3018–3025. SAGE Publications 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2023a. Cited on page 21.

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *IEEE transactions on pattern analysis and machine intelligence*, 46(6):4115–4128, 2024. Cited on page 20.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022. Cited on page 20.

Qinsheng Zhang, Molei Tao, and Yongxin Chen. gddim: Generalized denoising diffusion implicit models. In *The Eleventh International Conference on Learning Representations*, 2023b. Cited on page 5.

Yixian Zhang, Shu-ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. Sac flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *International Conference on Learning Representations*, volume 2026, pp. 121983–122016, 2026b. Cited on page 21.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. Cited on page 8.

Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024. Cited on page 21.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. *Advances in Neural Information Processing Systems*, 36: 55502–55542, 2023. Cited on page 20.

Guangyao Zhou, Sivaramakrishnan Swaminathan, Rajkumar Vasudeva Raju, J Swaroop Guntupalli, Wolfgang Lehrach, Joseph Ortiz, Antoine Dedieu, Miguel Lázaro-Gredilla, and Kevin Murphy. Diffusion model predictive control. *arXiv preprint arXiv:2410.05364*, 2024. Cited on page 20.

Appendix

TABLE OF CONTENTS

A Related Works **20**

B Theoretical Details **21**

B.1 Notations and Definitions 21

B.2 Useful Lemmas 22

B.3 Forward and Reverse Processes 24

B.4 Evidence Lower Bound Objective 25

B.5 Extension to DDIM and Guided Sampling 27

B.6 Understanding Why the Proposal Excels 33

C Experimental Setup **35**

C.1 Environments and Datasets 36

C.2 Baselines 38

C.3 Metrics 39

C.4 Physical Testbed 41

C.5 Detailed Implementation of **KDDM** 41

D Additional Experiments **42**

D.1 Generative Trajectory Stitching 42

D.2 More Image-Based Tasks 43

D.3 Effect of Kuramoto Model 43

D.4 Effect of Learning-Based $\mathbf{K}_\psi(\boldsymbol{\theta}_0, t)$ 44

D.5 Impact of Sampling Steps 44

E Visualizations **45**

E.1 (De)Synchronization Processes 45

E.2 Learned Distributions 45

E.3 Examples of Simulational Manipulation 48

E.4 Examples of Real-World Manipulation 50

F Limitations **52**

G Future Works **52**

A Related Works

Diffusion Paradigms for Data Generation. In recent years, diffusion models (Ho et al., 2020; Song et al., 2020b) have emerged as reliable and powerful deep generative models. We examine three interconnected views of diffusion modeling (Lai et al., 2025), each describing generation through a forward process that progressively perturbs data with noise and a reverse-time process approximated by a sequence of models that gradually denoise the perturbed samples. (i) Variational view: Originating from Variational Autoencoders (VAEs, Kingma & Welling, 2013), this view formulates diffusion as learning a progressive denoising process through a variational objective, giving rise to DDPM (Sohl-Dickstein et al., 2015; Ho et al., 2020) and its variants (Nichol & Dhariwal, 2021; Huang et al., 2021; Rissanen et al., 2022; Hoogeboom & Salimans, 2022; Yang et al., 2024). (ii) Score-Based view: Rooted in Energy-Based Models (EBMs, Ackley et al., 1985) and developed through Noise Conditional Score Networks (Song & Ermon, 2019), this view learns the score function, namely the gradient of the log data density, which provides the direction for progressively removing noise from samples. It further connects diffusion modeling with classical differential equation theory, offering a principled mathematical basis for analysis and algorithm design (Song et al., 2020b; Kawar et al., 2021; Karras et al., 2022; Daras et al., 2022; Kawar et al., 2022; Song et al., 2026). (iii) Flow-Based view: Building on Normalizing Flows (Rezende & Mohamed, 2015) and further developed through Flow Matching (FM) framework (Lipman et al., 2022; 2024; Tong et al., 2023) and Rectified Flow (Liu et al., 2023), this view formulates generation as a continuous transformation that transports samples from a simple prior toward the data distribution. The evolution is governed by a velocity field through an ODE, which explicitly characterizes the movement of probability mass over time. In this paper, we formulate **KDDM** from the variational view, whereas its closest counterpart, KODM, is developed from the score-based view. Nevertheless, as shown in Appendix B.5.2, there is an equivalent formulation from the second perspective, establishing a direct connection between the two views. As for the flow-based view, we leave it for future work.

Acceleration and Guidance of Diffusion Models. Diffusion models can be slow to sample, as generation typically requires hundreds of steps. DDIM (Song et al., 2020a) accelerates DDPM while preserving the same marginal distributions through a non-Markovian forward process, corresponding to a first-order exponential-Euler discretization from the ODE perspective. DEIS (Zhang & Chen, 2022) further uses higher-order Lagrange interpolation of past evaluations to approximate the next-step integral. The DPM-Solver family, including DPM-Solver (Lu et al., 2022), DPM-Solver++ (Lu et al., 2025a), and DPM-Solver-v3 (Zheng et al., 2023), exploits the semilinear structure of the Probability Flow ODE (PF-ODE) with a half-log SNR reparameterization, enabling high-quality sampling in around 10 steps. Many applications, however, require conditional generation, where outputs satisfy user-specified criteria. This can be achieved either by steering an unconditional model or by directly learning the conditional distribution. Classifier guidance (CG, Dhariwal & Nichol, 2021) steers generation with a pre-trained time-dependent classifier, while classifier-free guidance (CFG, Ho & Salimans, 2022) encodes conditions directly into the diffusion model. For training-free conditional generation, most methods leverage the predicted clean sample and use gradients from an off-the-shelf target predictor to guide the generation process (Li et al., 2022; Chung et al., 2022; 2023; Song et al., 2023; He et al., 2024; Ye et al., 2024). Other PnP approaches instead perform data-consistency updates (Lugmayr et al., 2022; Ho et al., 2022; Wang et al., 2022a) or use Sequential Monte Carlo (Trippe et al., 2023; Cardoso et al., 2024). Similarly, we also extend **KDDM** to accommodate DDIM (i.e., **KDIM**), CG, CFG, and a series of other guidance strategies.

Diffusion Models in Decision Making Scenarios. Diffusion models have recently been applied to a wide range of decision-making and control problems (Dong et al., 2024b; Wolf et al., 2025; Song et al., 2025). In what follows, we organize the discussion according to the three different roles mentioned in §3. ***The first*** is trajectory-level planners for long-horizon tasks (Chen et al., 2024b; Dong et al., 2024a; Zhou et al., 2024; Xiao et al., 2025; Balim et al., 2025; Zhang et al., 2026a). Initially, Diffuser (Janner et al., 2022) formulates planning as conditional trajectory generation and enables flexible behavior synthesis through CG. Other diffusion-based planners using CG include AdaptDiffuser (Liang et al., 2023) and MotionDiffuser (Zhang et al., 2024). With CFG, the planner can condition the diffusion process on a desired reward or value function. Examples include Decision Diffuser (Ajay et al., 2023), HDMI (Li et al., 2023), and Planning-as-In-Painting (Yang et al., 2023). Beyond CG and CFG, Diffusion Veteran (Lu et al., 2025b) and Prior Guidance (Ki et al., 2026) perform unconditional Monte Carlo sampling followed by selection, showing that this simple strategy can outperform CG and CFG when the dataset contains sufficient near-optimal trajectories. ***The second*** line of work

explores diffusion policies (Chen et al., 2022; Wang et al., 2022b; Pearce et al., 2023; Hansen-Estruch et al., 2023; Chi et al., 2025; Wagenmaker et al., 2025) as alternatives to deterministic or Gaussian policies, such as those used in PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018). In RL, Q-Score Matching (QSM) Psenka et al., 2023 laid the theoretical groundwork for this direction by establishing a connection between score functions and Q-value gradients. Building on this connection, subsequent studies have developed increasingly capable diffusion-based RL methods (Ding et al., 2024; Li et al., 2024b; Dong et al., 2025; Celik et al., 2025; Lyu et al., 2026; Zhang et al., 2026b). In IL, diffusion-based behavior cloning (BC) has gained considerable traction in the control and robotics communities, achieving SOTA performance in real-world robotic tasks ranging from single-task applications (Ankile et al., 2024; Zhao et al., 2024; Sridhar et al., 2024; Dasari et al., 2025) to "generalist" frameworks (Team et al., 2024; Black et al., 2024; Intelligence et al., 2025b; Bjorck et al., 2025; Intelligence et al., 2026; Cadene et al., 2026). Recent efforts have increasingly emphasized expressive network architectures that enable effective guided sampling from rich multimodal agent observations, spanning low-dimensional physical states (Pearce et al., 2023), RGB images (Chi et al., 2025), 3D point clouds (Ze et al., 2024), and even natural-language instructions (Bell et al., 2024). Unlike planners and policies, ***the third*** role, data synthesis with DMs, remains a relatively underexplored direction. Nevertheless, it has shown promise in augmenting policy learning and mitigating the limited fidelity of existing datasets (Imre, 2021; Cho et al., 2022) by synthesizing additional data, ranging from individual transitions to complete trajectories (Lu et al., 2023; Zhang et al., 2023a; Li et al., 2024a; Yang & Wang, 2025). Despite the momentum of these studies, most of them simply plug classical diffusion paradigms, such as DDPM and Flow Matching, into the backbone. Yet, these existing approaches generally overlook spatio-temporal dependencies (see §3 in the main text), which are crucial to real-world decision-making. This gap lies at the heart of our motivation for developing **KDDM** tailored for decision-making tasks. Further, we show that **KDDM** can seamlessly accommodate all three roles discussed above while supporting nearly all of their expected capabilities.

B Theoretical Details

In this section, we provide the detailed theoretical proofs and analyses that underpin the main results presented in the paper. Some key inspirations were excerpted from prior works of Sohl-Dickstein et al. (2015); Song & Ermon (2019); Chopin et al. (2020); Ho et al. (2020); Yang et al. (2024); Lai et al. (2025); Song et al. (2026).

B.1 Notations and Definitions

Definition 1 (Lipschitz Continuity). *Let $C \subset \mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is said to be ρ-Lipschitz on C if there exists a constant $\rho \geq 0$ such that $\|f(w_1) - f(w_2)\| \leq \rho\|w_1 - w_2\|$.*

Definition 2 (Kullback–Leibler Divergence). *For two probability distributions p and q over $\mathcal{X}$, the Kullback–Leibler Divergence (KLD) is defined as $\mathcal{D}_{\mathrm{KL}}(p \parallel q) = \sum_{x\in\mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$. If the two distributions $P = \mathcal{N}(\mu_1, \Sigma_1)$ and $Q = \mathcal{N}(\mu_2, \Sigma_2)$, the KLD can be computed as*

$$\mathcal{D}_{\mathrm{KL}}(P \parallel Q) = \frac{1}{2}\left[\mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) - d + \ln\left(\frac{\det \Sigma_2}{\det \Sigma_1}\right)\right],$$

where d is the dimensionality of the target data.

Definition 3 (Evidence Lower Bound). *For any data sample $\mathbf{x}_0$, the log-likelihood is bounded by the Evidence Lower Bound (ELBO):*

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0)] &= \mathbb{E}_{q(\mathbf{x}_0)}\left[\log\left(\int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T}\right)\right] \\
&= \mathbb{E}_{q(\mathbf{x}_0)}\left[\log\left(\int q(\mathbf{x}_{1:T}|\mathbf{x}_0)\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T}\right)\right] \\
&= \mathbb{E}_{q(\mathbf{x}_0)}\left[\log\left(\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]\right)\right] \qquad \triangleright \textit{Using Jensen's inequality} \\
&\geq \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = -\mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log\frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p(\mathbf{x}_{0:T})}\right] := \mathrm{ELBO}
\end{aligned}$$

Definition 4 (Feynman-Kac Models). *Consider a model starts from a Markov probability law on a state space $\mathcal{X}$, defined by an initial distribution $\mathbb{M}_0$ and transition kernels $\mathbb{M}_{1:T}$: $\mathbb{M}_T(dx_{0:T}) = \mathbb{M}_0(dx_0)\prod_{t=1}^{T}\mathbb{M}_t(x_{t-1}, dx_t)$. Let $G_0 : \mathcal{X} \to \mathbb{R}^+$ and $G_t : \mathcal{X}^2 \to \mathbb{R}^+$ for $1 \le t \le T$ be potential functions. The associated Feynman–Kac measures $\{\mathbb{Q}_t\}_{t=0}^{T}$ are defined as the change of measure*

$$\mathbb{Q}_t(dx_{0:t}) = \frac{1}{L_t} G_0(x_0) \prod_{s=1}^{t} G_s(x_{s-1}, x_s)\, \mathbb{M}_t(dx_{0:t}),$$

where L_t is the normalizing constant ensuring that $\mathbb{Q}_t$ is a probability measure.

B.2 Useful Lemmas

Lemma 1. *If a non-negative function $u(t)$ satisfies $u'(t) \le au(t)$ for $t \ge 0$, then $u(t) \le u(0)e^{at}$.*

Proof. Consider $w(t) = u(t)e^{-at}$, then we have $w' = e^{-at}(u' - au) \le 0$. So apparently, $w(t) \le w(0) \Rightarrow u(t)e^{-at} \le u(0)$. Finally, $u(t) \le u(0)e^{at}$. □

Lemma 2. *Given a marginal Gaussian distribution for $\mathbf{x}$ and a conditional Gaussian distribution for $\mathbf{y}$ given $\mathbf{x}$ in the form: $\mathbf{x} \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right)$, $\mathbf{y}|\mathbf{x} \sim \mathcal{N}\left(\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right)$, the marginal distribution of $\mathbf{y}$ is given by $p(\mathbf{y}) = \int_{\mathbf{x}} p(\mathbf{x})\, p(\mathbf{y}|\mathbf{x})\, d\mathbf{x} = \mathcal{N}\left(\mathbf{y} \mid \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\top}\right)$.*

Proof. This is a well-known property of marginal and conditional Gaussian distributions; see (Bishop & Nasrabadi, 2006, Section 2.3, page 93). □

Lemma 3. *Consider a joint Gaussian distribution that*

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_a \\ \mathbf{x}_b\end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{\mu} = \begin{bmatrix}\boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b\end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix}\boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb}\end{bmatrix}\right), \quad \boldsymbol{\Lambda} = \begin{bmatrix}\boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb}\end{bmatrix}$$

where $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$. Then the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is Gaussian with conditional mean $\mathbb{E}[\mathbf{x}_a|\mathbf{x}_b] = bmu_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b) = \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b)$.

Proof. Also, it is provided in (Bishop & Nasrabadi, 2006, Section 2.3, page 90). □

Lemma 4. *Assume that the potential function G_t is bounded. Let $(X_t^n, W_t^n)_{n=1}^N$ denote the particle system generated by a multinomial resampling-based sequential Monte Carlo algorithm at $t \ge 0$. For any test function $f \in \mathcal{C}_b(\mathcal{X})$ such that $f \times G_t \in \mathcal{C}_b(\mathcal{X})$, $\sum_{n=1}^N W_t^n f(X_t^n) \xrightarrow[N\to\infty]{a.s.} \mathbb{Q}_t(f)$.*

Proof. This theoretical result of the sequential Monte Carlo algorithm follows directly from Proposition 11.4 in (Chopin et al., 2020, Section 11.2, page 173). □

Lemma 5 (Mean-Field Reduction). *Consider the order parameter $re^{i\psi} = \frac{1}{N}\sum_{m=1}^N e^{iz_m}$. The pairwise sinusoidal coupling over the entire population can be reduced to a single-body interaction with the mean field, satisfying: $\frac{1}{N}\sum_{m=1}^N \sin(z_m - z_n) = r\sin(\psi - z_n)$, $\forall n \in \{1, \ldots, N\}$.*

Proof. By definitions and $e^{i(z_m - z_n)} = e^{-iz_n}e^{iz_m}$, we can write

$$\frac{1}{N}\sum_{m=1}^{N}\sin(z_m - z_n) = \mathbf{Im}\left(\frac{1}{N}\sum_{m=1}^{N} e^{i(z_m - z_n)}\right) = \mathbf{Im}\left(e^{-iz_n}\left(\frac{1}{N}\sum_{m=1}^{N} e^{iz_m}\right)\right) = \mathbf{Im}\left(re^{i(\psi - z_n)}\right),$$

where $\mathbf{Im}$ denotes the imaginary part of a complex number. Applying Euler's formula, i.e., $e^{i\theta} = \cos\theta + i\sin\theta$, we obtain

$$re^{i(\psi - z_n)} = r\cos(\psi - z_n) + ir\sin(\psi - z_n) \Rightarrow \mathbf{Im}\left(re^{i(\psi - z_n)}\right) = r\sin(\psi - z_n).$$

Therefore, $\frac{1}{N}\sum_{m=1}^{N}\sin(z_m - z_n) = r\sin(\psi - z_n)$, which concludes the proof. □

Lemma 6 (Lipschitzness of Couplings). *Let $\mathbf{K}_\psi(\cdot, t) : \mathbb{R}^N \to \mathbb{R}^N$ denote the flow map associated with the Kuramoto dynamics $\frac{dz_t}{dt} = \frac{K}{N}\sum_{m=1}^{N}\sin(z_t^m - z_t^n)$. That is, $\mathbf{K}_\psi(z_0, t)$ represents the state at time t starting from the initial condition z_0. Then, for any $t \geq 0$ and any $\tilde{z}_0, \hat{z}_0 \in \mathbb{R}^N$, $\|\mathbf{K}_\psi(\tilde{z}_0, t) - \mathbf{K}_\psi(\hat{z}_0, t)\| \leq \rho_t\|\tilde{z}_0 - \hat{z}_0\|$, where $\rho_t = e^{2Kt}$.*

Proof. We first define a function $F_n(x) = \frac{K}{N}\sum_m \sin(x_m - x_n)$, whose Jacobian $\mathbf{J}_{F_n}$ satisfies

$$\frac{\partial F_n}{\partial x_j} = \begin{cases} \frac{K}{N}\cos(x_j - x_n), & j \neq n, \\ -\frac{K}{N}\sum_{m\neq n}\cos(x_m - x_n), & j = n. \end{cases}$$

Due to $|\cos(\cdot)| \leq 1$, we can write $\sum_j \left|\frac{\partial F_n}{\partial x_j}\right| \leq \frac{K}{N}(N-1) + \frac{K}{N}(N-1) = 2K\left(1 - \frac{1}{N}\right)$, which also means $\|\mathbf{J}_F\|_\infty \leq 2K$ where $F(x) = [F_1(x), \ldots, F_N(x)]^\top$. Now using the Chain rule and according to the Mean value theorem (under ∞-norm), we obtain

$$\|F(x) - F(y)\| = \left\|\int_0^1 \mathbf{J}_F\left(y + s(x-y)\right)(x-y)\,ds\right\| \leq \int_0^1 \|\mathbf{J}_F\|_\infty\|x-y\|ds \leq 2K\|x-y\|.$$

Next, we set $x(t) = \mathbf{K}_\psi(\tilde{z}_0, t)$, $y(t) = \mathbf{K}_\psi(\hat{z}_0, t)$, and define the error $\mathbf{e}(t) \coloneqq x(t) - y(t)$. Clearly, $\dot{\mathbf{e}} = \frac{d\mathbf{e}(t)}{dt} = F(x(t)) - F(y(t))$. Because the Lipschitz constant of $F(x)$ is $2K$, the inequality $\|\dot{\mathbf{e}}\| \leq 2K\|\mathbf{e}(t)\|$ holds. Combining it with Lemma 1, we have $\|\mathbf{e}(t)\| \leq e^{2Kt}\|\mathbf{e}(0)\|$. Therefore, $\|\mathbf{K}_\psi(\tilde{z}_0, t) - \mathbf{K}_\psi(\hat{z}_0, t)\| \leq \rho_t\|\tilde{z}_0 - \hat{z}_0\|$ where $\rho_t = e^{2Kt}$, which concludes the proof. □

Lemma 7 (Asymptotic Synchronization). *Consider a Kuramoto model governed by the dynamics $\frac{dz_n}{dt} = \frac{K}{N}\sum_{m=1}^{N}\sin(z_m - z_n)$. Then, as $t \to \infty$, the phase velocities asymptotically converge to zero, i.e., $\frac{dz_n}{dt} \to \mathbf{0}$.*

Proof. From Lemma 5, we know that $\frac{1}{N}\sum_{m=1}^{N}\sin(z_m - z_n) = r\sin(\psi - z_n)$. Then the non-linear system can be written as $\dot{z}_n = \frac{dz_n}{dt} = Kr\sin(\psi - z_n)$. If $K > 0$, the system ultimately converges to a synchronized state: $z_1 = z_2 = \cdots = z_N \pmod{2\pi}$. In this synchronized state, we have $r = 1$ and $\psi = z_n$. Consequently, $\dot{z}_n = K\sin(0) = 0$, which implies $\dot{z}_n \to 0$.

More formally, by defining $\delta_{mn} \coloneqq z_m - z_n$, we now turn to proving that the phase differences $|\max_n z_n - \min_n z_n|$ are non-increasing over time. Without loss of generality, let $z_{\max} = z_i$, then we have $\dot{z}_i = \frac{K}{N}\sum_{m=1}^{N}\sin(z_m - z_i) \leq 0$. Similarly, if $z_{\min} = z_j$, then $\dot{z}_j \geq 0$. As a result, the phase diameter, defined as $D(t) \coloneqq |\max_n z_n - \min_n z_n|$, is monotonically decreasing over time. Furthermore, when $K > 0$ and the initial diameter satisfies $D(0) < \pi$, it can be further shown that $D(t) \to \mathbf{0}$. Consequently, $z_m - z_n \to \mathbf{0}$, which implies $\sin(z_m - z_n) \to \mathbf{0}$. Ultimately, the dynamics of each oscillator satisfy $\frac{dz_n}{dt} = \frac{K}{N}\sum_{m=1}^{N}\sin(z_m - z_n) \to \mathbf{0}$. □

Lemma 8 (Tweedie's Formula). *Given a prior distribution $\boldsymbol{\eta} \sim g(\cdot)$, suppose that the distribution $p(\mathbf{x}|\boldsymbol{\eta})$ belongs to the exponential family: $p(\mathbf{x}|\boldsymbol{\eta}) = p_0(\mathbf{x})\exp\left(\boldsymbol{\eta}^\top F(\mathbf{x}) - L(\boldsymbol{\eta})\right)$. Then the posterior mean $\hat{\boldsymbol{\eta}}$ of $p(\boldsymbol{\eta}|\mathbf{x})$ uniquely satisfies $(\nabla_{\mathbf{x}}F(\mathbf{x}))^\top\hat{\boldsymbol{\eta}} = \nabla_{\mathbf{x}}\log p(\mathbf{x}) - \nabla_{\mathbf{x}}\log p_0(\mathbf{x})$ where $F(\mathbf{x})$ denotes the sufficient statistic, $\boldsymbol{\eta}$ is the natural (canonical) parameter, $L(\boldsymbol{\eta})$ is the log-partition (cumulant generating) function ensuring that $p(\mathbf{x}|\boldsymbol{\eta})$ is properly normalized, and $p_0(\mathbf{x})$ is the base measure corresponding to $\boldsymbol{\eta} = \mathbf{0}$.*

Proof. We first compute the derivative of the marginal distribution $p(\mathbf{x})$ with respect to $\mathbf{x}$ as

$$\nabla_{\mathbf{x}}p(\mathbf{x}) = \nabla_{\mathbf{x}}\int p(\mathbf{x}|\boldsymbol{\eta})\,g(\boldsymbol{\eta})\,d\boldsymbol{\eta} = \nabla_{\mathbf{x}}\int p_0(\mathbf{x})\exp\left(\boldsymbol{\eta}^\top F(\mathbf{x}) - L(\boldsymbol{\eta})\right)g(\boldsymbol{\eta})\,d\boldsymbol{\eta}$$

$$
\begin{aligned}
&= \left(\nabla_{\mathbf{x}} F(\mathbf{x})\right)^{\top} \int \boldsymbol{\eta} p_0(\mathbf{x}) \exp\left(\boldsymbol{\eta}^{\top} F(\mathbf{x}) - L(\boldsymbol{\eta})\right) g(\boldsymbol{\eta})\, d\boldsymbol{\eta} \\
&\quad + \nabla_{\mathbf{x}} p_0(\mathbf{x}) \int \exp\left(\boldsymbol{\eta}^{\top} F(\mathbf{x}) - L(\boldsymbol{\eta})\right) g(\boldsymbol{\eta})\, d\boldsymbol{\eta} \\
&= \left(\nabla_x F(\mathbf{x})\right)^{\top} \int \boldsymbol{\eta} p(\mathbf{x}, \boldsymbol{\eta}) d\boldsymbol{\eta} + \frac{\nabla_{\mathbf{x}} p_0(\mathbf{x})}{p_0(\mathbf{x})} \int p(\mathbf{x}|\boldsymbol{\eta})\, g(\boldsymbol{\eta})\, d\boldsymbol{\eta} \\
&= \left(\nabla_x F(\mathbf{x})\right)^{\top} \int \boldsymbol{\eta} p(\mathbf{x}, \boldsymbol{\eta}) d\boldsymbol{\eta} + \frac{\nabla_{\mathbf{x}} p_0(\mathbf{x})}{p_0(\mathbf{x})} p(\mathbf{x}).
\end{aligned}
$$

As a consequence, $\left(\nabla_{\mathbf{x}} F(\mathbf{x})\right)^{\top} \int \boldsymbol{\eta} p(\boldsymbol{\eta}|\mathbf{x})\, d\boldsymbol{\eta} = \frac{\nabla_{\mathbf{x}} p(\mathbf{x})}{p(\mathbf{x})} - \frac{\nabla_{\mathbf{x}} p_0(\mathbf{x})}{p_0(\mathbf{x})}$. Then we have $\left(\nabla_{\mathbf{x}} F(\mathbf{x})\right)^{\top} \hat{\boldsymbol{\eta}} = \nabla_{\mathbf{x}} \log p(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_0(\mathbf{x})$, which concludes the proof. □

B.3 Forward and Reverse Processes

Starting from the forward transition kernel $q_\psi(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_0)$ in Eqn. (4), we establish Theorem 1, which provides explicit closed-form expressions for $q_\psi(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0)$ and $q_\psi(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0)$.

Theorem 1. *Consider the forward diffusion process of* ***KDDM*** *defined in Eqn. (4), then the forward marginal distribution admits the closed-form expression*

$$
q_\psi(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t), (1 - \bar{\alpha}_t)\mathbf{I}\right).
$$

Moreover, the reverse distributions conditioned on $\boldsymbol{\theta}_0$ are also Gaussians: $q_\psi(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0) = \mathcal{N}\left(\tilde{\boldsymbol{\mu}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0), \tilde{\beta}_t \mathbf{I}\right)$, where variance $\tilde{\beta}_t = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}$, and mean

$$
\tilde{\boldsymbol{\mu}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\left(\boldsymbol{\theta}_t - \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t)\right) + \kappa_{t-1}\mathbf{K}_\psi(\boldsymbol{\theta}_0, t-1) + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}\boldsymbol{\theta}_0.
$$

Proof. Using the reparameterization trick (Kingma & Welling, 2013), we can directly sample $\boldsymbol{\theta}_t$ at any arbitrary diffusion step t as

$$
\begin{aligned}
\boldsymbol{\theta}_t &= \sqrt{\alpha_t}\boldsymbol{\theta}_{t-1} + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t) - \sqrt{\alpha_t}\kappa_{t-1}\mathbf{K}_\psi(\boldsymbol{\theta}_0, t-1) + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{\theta}_{t-2} + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t) - \sqrt{\alpha_{t-1}}\kappa_{t-2}\mathbf{K}_\psi(\boldsymbol{\theta}_0, t-2) + \sqrt{1 - \alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}\boldsymbol{\theta}_{t-3} + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t) - \sqrt{\alpha_{t-2}}\kappa_{t-3}\mathbf{K}_\psi(\boldsymbol{\theta}_0, t-3) + \sqrt{1 - \alpha_t\alpha_{t-1}\alpha_{t-2}}\bar{\boldsymbol{\epsilon}}_{t-3} \\
&= \cdots = \sqrt{\alpha_t\alpha_{t-1}\cdots\alpha_1}\boldsymbol{\theta}_0 + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t) - 0 + \sqrt{1 - \alpha_t\alpha_{t-1}\cdots\alpha_1}\bar{\boldsymbol{\epsilon}}_0 \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t) + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon} \qquad \triangleright \text{ Here } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})
\end{aligned}
$$

where $\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\bar{\boldsymbol{\epsilon}}_{t-2}, \bar{\boldsymbol{\epsilon}}_{t-3}, \ldots$ denote standard Gaussian noise variables obtained by combining multiple independent Gaussian terms. The above equalities follow from repeatedly applying the transition kernel and repeatedly combining independent Gaussian noise terms, whose variances add under linear combinations. For instance, the first merged standard deviation is $\sqrt{(1 - \alpha_t) + \alpha_t(1 - \alpha_{t-1})} = \sqrt{1 - \alpha_t\alpha_{t-1}}$. Therefore, the marginal distribution of $\boldsymbol{\theta}_t$ conditioned on $\boldsymbol{\theta}_0$ can be expressed as $q_\psi(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t), (1 - \bar{\alpha}_t)\mathbf{I}\right)$.

We next focus on the reverse conditional probability $q_\psi(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t)$, which is tractable because it is also Gaussian when conditioned on $\boldsymbol{\theta}_0$: $q_\psi(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0) = \mathcal{N}\left(\boldsymbol{\theta}_{t-1}; \tilde{\boldsymbol{\mu}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0), \tilde{\boldsymbol{\Sigma}}_t \mathbf{I}\right)$. Using the Bayes' rule, we have

$$
\begin{aligned}
q_\psi(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0) &= q_\psi(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_0) \frac{q_\psi(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0)}{q_\psi(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0)} \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{\left(\boldsymbol{\theta}_t - \sqrt{\alpha_t}\boldsymbol{\theta}_{t-1} - \mathbf{K}_t + \sqrt{\alpha_t}\mathbf{K}_{t-1}\right)^2}{\beta_t}\right.\right. \qquad \triangleright \text{ For simplicity, let } \mathbf{K}_t \coloneqq \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t) \\
&\qquad \left.\left. + \frac{\left(\boldsymbol{\theta}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 - \mathbf{K}_{t-1}\right)^2}{1 - \bar{\alpha}_{t-1}} - \frac{\left(\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 - \mathbf{K}_t\right)^2}{1 - \bar{\alpha}_t}\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \exp\left(-\frac{1}{2}\left(\frac{\alpha_t\boldsymbol{\theta}_{t-1}^2 + 2\left(\sqrt{\alpha_t}\mathbf{K}_t - \alpha_t\mathbf{K}_{t-1} - \sqrt{\alpha_t}\boldsymbol{\theta}_t\right)\boldsymbol{\theta}_{t-1}}{\beta_t}\right.\right. \qquad \triangleright\ \mathbf{C}\left(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0, t\right) \text{ involves no } \boldsymbol{\theta}_{t-1} \\
&\qquad\qquad \left.\left. + \frac{\boldsymbol{\theta}_{t-1}^2 - 2\left(\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{K}_{t-1}\right)\boldsymbol{\theta}_{t-1}}{1-\bar{\alpha}_{t-1}}\right) + \mathbf{C}\left(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0, t\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(-2\left(\frac{-\sqrt{\alpha_t}\mathbf{K}_t + \alpha_t\mathbf{K}_{t-1} + \sqrt{\alpha_t}\boldsymbol{\theta}_t}{\beta_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{K}_{t-1}}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{\theta}_{t-1}\right.\right. \\
&\qquad\qquad \left.\left. + \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{\theta}_{t-1}^2\right) + \mathbf{C}\left(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0, t\right)\right)
\end{aligned}
$$

where the terms collected in $\mathbf{C}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0, t)$ are independent of $\boldsymbol{\theta}_{t-1}$ and can therefore be absorbed into the normalization constant. By extracting the coefficient of the quadratic term above (also recall that $\beta_t = 1 - \alpha_t$), we can respectively compute the variance and mean as

$$
\begin{aligned}
\tilde{\boldsymbol{\Sigma}}_t \coloneqq \tilde{\beta}_t &= \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)^{-1} = \left(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\left(1-\bar{\alpha}_{t-1}\right)\beta_t}\right)^{-1} = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t} \\
\tilde{\boldsymbol{\mu}}\left(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right) &= \left(\frac{-\sqrt{\alpha_t}\mathbf{K}_t + \alpha_t\mathbf{K}_{t-1} + \sqrt{\alpha_t}\boldsymbol{\theta}_t}{\beta_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{K}_{t-1}}{1-\bar{\alpha}_{t-1}}\right) \Big/ \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) \\
&= \left(\frac{-\sqrt{\alpha_t}\mathbf{K}_t + \alpha_t\mathbf{K}_{t-1} + \sqrt{\alpha_t}\boldsymbol{\theta}_t}{\beta_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{K}_{t-1}}{1-\bar{\alpha}_{t-1}}\right) \cdot \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t} \\
&= \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\left(\boldsymbol{\theta}_t - \mathbf{K}_t\right) + \mathbf{K}_{t-1} + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{\theta}_0
\end{aligned}
$$

Hence, the proof of Theorem 1 is completed. □

B.4 Evidence Lower Bound Objective

Below is a derivation of Eqn. (8) as in Theorem 2, the reduced variance variational bound or/and ELBO for our proposed diffusion models.

Theorem 2. *Let $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\boldsymbol{\theta}_0 \sim q_{\text{data}}$. Then, under the parameterization in Eqn. (6), the Evidence Lower Bound (ELBO) of* ***KDDM*** *satisfies*

$$
-\text{ELBO} \leq \mathbf{C} + \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\theta}_0, \boldsymbol{\epsilon}}\left[k_t \left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t \mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right) + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, t\right)\right\|_2^2\right]
$$

for some constants $\mathbf{C}$ and t-related coefficients k_t.

Proof. We start by expanding the ELBO in Eqn.(7) into the sum of a sequence of tractable KLDs:

$$
\begin{aligned}
\text{ELBO} &= -\mathbb{E}_{q_\psi}\left[\log \frac{q_\psi\left(\boldsymbol{\theta}_{1:T}|\boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_{0:T}\right)}\right] = -\mathbb{E}_{q_\psi}\left[\log \frac{\prod_{t=1}^{T} q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}\right)}{p\left(\boldsymbol{\theta}_T\right)\prod_{t=1}^{T} p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)}\right] \\
&= -\mathbb{E}_{q_\psi}\left[-\log p\left(\boldsymbol{\theta}_T\right) + \sum_{t=2}^{T}\log\frac{q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}\right)}{p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)} + \log\frac{q_\psi\left(\boldsymbol{\theta}_1|\boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_1\right)}\right] \\
&= -\mathbb{E}_{q_\psi}\left[-\log p\left(\boldsymbol{\theta}_T\right) + \sum_{t=2}^{T}\log\frac{q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right) q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right) q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0\right)} + \log\frac{q_\psi\left(\boldsymbol{\theta}_1|\boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_1\right)}\right] \\
&= -\mathbb{E}_{q_\psi}\left[-\log p\left(\boldsymbol{\theta}_T\right) + \sum_{t=2}^{T}\log\frac{q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)} + \sum_{t=2}^{T}\log\frac{q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)}{q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0\right)} + \log\frac{q_\psi\left(\boldsymbol{\theta}_1|\boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_1\right)}\right] \\
&= -\mathbb{E}_{q_\psi}\left[-\log p\left(\boldsymbol{\theta}_T\right) + \sum_{t=2}^{T}\log\frac{q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)} + \log\frac{q_\psi\left(\boldsymbol{\theta}_T|\boldsymbol{\theta}_0\right)}{q_\psi\left(\boldsymbol{\theta}_1|\boldsymbol{\theta}_0\right)} + \log\frac{q_\psi\left(\boldsymbol{\theta}_1|\boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_1\right)}\right]
\end{aligned}
$$

$$= -\mathbb{E}_{q_\psi}\left[\log \frac{q_\psi\left(\boldsymbol{\theta}_T|\boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_T\right)} + \sum_{t=2}^{T} \log \frac{q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right)}{p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)} - \log p\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_1\right)\right]$$

$$= -\mathbb{E}_{q_\psi}\left[\mathcal{D}_{\mathrm{KL}}\left(q_\psi\left(\boldsymbol{\theta}_T|\boldsymbol{\theta}_0\right) \parallel p\left(\boldsymbol{\theta}_T\right)\right) + \sum_{t=2}^{T} \mathcal{D}_{\mathrm{KL}}\left(q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right) \parallel p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)\right) - \log p\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_1\right)\right]$$

Next, we calculate each term of the ELBO expansion. According to Lemma 7, $\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, T\right) \to \mathbf{0}$ for sufficiently large T, then based on Definition 2, the first constant term which can be ignored during learning is

$$\begin{aligned}
\mathbb{E}_{q_\psi}\left[\mathcal{D}_{\mathrm{KL}}\left(q_\psi\left(\boldsymbol{\theta}_T|\boldsymbol{\theta}_0\right) \parallel p\left(\boldsymbol{\theta}_T\right)\right)\right] &\approx \mathbb{E}_{\boldsymbol{\theta}_0}\left[\mathcal{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\sqrt{\bar{\alpha}_T}\boldsymbol{\theta}_0 + \kappa_t \cdot \mathbf{0}, \left(1-\bar{\alpha}_T\right)\mathbf{I}\right) \parallel \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)\right)\right] \\
&= \frac{1}{2}\mathbb{E}_{\boldsymbol{\theta}_0}\left[\left\|\sqrt{\bar{\alpha}_T}\boldsymbol{\theta}_0 - \mathbf{0}\right\|_2^2\right] + d\left(\log \frac{1}{\sqrt{1-\bar{\alpha}_T}} + \frac{1-\bar{\alpha}_T-1}{2}\right) \\
&= \frac{\bar{\alpha}_T}{2}\mathbb{E}_{\boldsymbol{\theta}_0}\left[\left\|\boldsymbol{\theta}_0\right\|_2^2\right] - \frac{d}{2}\left(\bar{\alpha}_T + \log\left(1-\bar{\alpha}_T\right)\right).
\end{aligned}$$

Regarding the calculation of $\mathcal{D}_{\mathrm{KL}}\left(q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right) \parallel p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)\right)$, the KL divergence between $q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right)$ and $p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)$ is $1/2\tilde{\beta}_t$ times the squred ℓ_2 distance between their means, because we assume that they have the same fixed covariance matrix $\tilde{\beta}_t\mathbf{I}$. We set $\hat{\boldsymbol{\theta}}_0 \coloneqq \hat{\boldsymbol{\theta}}_0\left(\boldsymbol{\theta}_t, t\right)$. Using the nice property of $q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)$, i.e., $\boldsymbol{\theta}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right) + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, we have

$$\begin{aligned}
&\mathbb{E}_{q_\psi}\left[\mathcal{D}_{\mathrm{KL}}\left(q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right) \parallel p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)\right)\right] = \mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\tilde{\beta}_t}\left\|\tilde{\boldsymbol{\mu}}\left(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right) - \hat{\boldsymbol{\mu}}\left(\boldsymbol{\theta}_t, t\right)\right\|_2^2\right] \\
&= \mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\tilde{\beta}_t}\left\|\frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\left(\boldsymbol{\theta}_t - \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)\right) + \kappa_{t-1}\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t-1\right) + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{\theta}_0\right.\right. \\
&\qquad \left.\left. - \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\left(\boldsymbol{\theta}_t - \kappa_t\mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t\right)\right) - \kappa_{t-1}\mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t-1\right) - \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{\boldsymbol{\theta}}_0\right\|_2^2\right] \\
&= \mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\tilde{\beta}_t}\left\|\kappa_{t-1}\left(\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t-1\right) - \mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t-1\right)\right) + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\left(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\right)\right.\right. \\
&\qquad \left.\left. + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\kappa_t}{1-\bar{\alpha}_t}\left(\mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t\right) - \mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)\right)\right\|_2^2\right] \\
&\leq \frac{1}{2\tilde{\beta}_t}\mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[\kappa_{t-1}\left\|\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t-1\right) - \mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t-1\right)\right\|_2^2 + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\right\|_2^2\right. \\
&\qquad \left. + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\kappa_t}{1-\bar{\alpha}_t}\left\|\mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t\right) - \mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)\right\|_2^2\right]
\end{aligned}$$

From Lemma 6, we know that $\left\|\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right) - \mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t\right)\right\|_2 \leq \rho_t\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\right\|_2$. Thus we continue by

$$\begin{aligned}
&\mathbb{E}_{q_\psi}\left[\mathcal{D}_{\mathrm{KL}}\left(q_\psi\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right) \parallel p\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)\right)\right] \\
&\leq \frac{1}{2\tilde{\beta}_t}\mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[\kappa_{t-1}\rho_{t-1}\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\right\|_2^2 + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\right\|_2^2 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\kappa_t\rho_t}{1-\bar{\alpha}_t}\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\right\|_2^2\right] \\
&= \mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[\frac{1-\bar{\alpha}_t}{2\left(1-\bar{\alpha}_{t-1}\right)\beta_t}\left(\kappa_{t-1}\rho_{t-1} + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\kappa_t\rho_t}{1-\bar{\alpha}_t}\right)\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\right\|_2^2\right] \\
&= \mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[k_t\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\right\|_2^2\right] \qquad \triangleright \text{ Here } k_t \coloneqq \frac{1}{2\tilde{\beta}_t}\left(\kappa_{t-1}\rho_{t-1} + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\kappa_t\rho_t}{1-\bar{\alpha}_t}\right)
\end{aligned}$$

Finally, given $\alpha_0 = 1$ and $\kappa_0 = 0$, we have

$$\mathbb{E}_{q_\psi}\left[\log p\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_1\right)\right] = \mathbb{E}_{q_\psi}\left[\log \mathcal{N}\left(\boldsymbol{\theta}_0; \frac{\beta_1}{1-\alpha_1}\hat{\boldsymbol{\theta}}_0\left(\boldsymbol{\theta}_1, 1\right), \boldsymbol{\Sigma}_1\mathbf{I}\right)\right]$$

$$= \mathbb{E}_{q_\psi}\left[\left(-\frac{d}{2}\log 2\pi\boldsymbol{\Sigma}_1 - \frac{1}{2\Sigma_1}\left\|\boldsymbol{\theta}_0 - \frac{\beta_1}{1-\alpha_1}\hat{\boldsymbol{\theta}}_0\left(\boldsymbol{\theta}_1, 1\right)\right\|_2^2\right)\right]$$
$$= -\frac{d}{2}\log 2\pi\boldsymbol{\Sigma}_1 - \mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\Sigma_1}\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\left(\boldsymbol{\theta}_1, 1\right)\right\|_2^2\right]$$

Putting the above together, the ELBO can be expressed as

$$-\text{ELBO} \leq \mathbf{C} + \sum_{t=2}^{T}\mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[k_t\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\left(\boldsymbol{\theta}_t, t\right)\right\|_2^2\right] + \mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[k_1\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\left(\boldsymbol{\theta}_1, 1\right)\right\|_2^2\right]$$
$$\leq \mathbf{C} + \sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{\theta}_0,\boldsymbol{\epsilon}}\left[k_t\left\|\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right) + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, t\right)\right\|_2^2\right]$$

where k_1 is defined as $1/2\boldsymbol{\Sigma}_1$. The computation of the variational lower bound loss is now finished. □

B.5 Extension to DDIM and Guided Sampling

As mentioned in the main text, a core advantage of **KDDM** over KODM lies in its greater sampling flexibility. The following exposition presents how to achieve fast and controllable sampling in **KDDM**.

B.5.1 Kuramoto Decision Implicit Model

To accelerate generative inference, we first extend our framework to ODEs, like Denoising Diffusion Implicit Models (DDIMs) which use a non-Markovian inference process defined as

$$q_\sigma\left(\mathbf{x}_{1:T}|\mathbf{x}_0\right) \coloneqq q_\sigma\left(\mathbf{x}_T|\mathbf{x}_0\right)\prod_{t=2}^{T}q_\sigma\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0\right), \tag{11}$$

where $q_\sigma\left(\mathbf{x}_T|\mathbf{x}_0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_T}\mathbf{x}_0, \left(1-\bar{\alpha}_T\right)\mathbf{I}\right)$, $\sigma \in \mathbb{R}_{\geq 0}^T$, and for all $t > 1$,

$$q_\sigma\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I}\right). \tag{12}$$

It has the same marginal (joint) distributions and training objective as the classical diffusion model (i.e., VE-SDE[1]), but it admits forward processes with lengths smaller than T in theory. Similarly, as derived in Proposition 1, one can also generate a sample $\boldsymbol{\theta}_{t-\Delta t}$ from a sample $\boldsymbol{\theta}_t$ via Eqn. (10).

Proposition 1 (Generalized DDIM Sampling). *Consider the forward diffusion process defined in Eqn. (4), and a variance schedule $\{\sigma_t\}_{t=1}^T$. Then for $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the DDIM reverse transition $q_\sigma(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \boldsymbol{\theta}_0)$ in **KDDM** can be parameterized as sampling*

$$\boldsymbol{\theta}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{\theta}}_0 + \kappa_{t-1}\mathbf{K}_\psi(\hat{\boldsymbol{\theta}}_0, t-1) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{\theta}}_0 - \kappa_t\mathbf{K}_\psi(\hat{\boldsymbol{\theta}}_0, t)}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t\boldsymbol{\epsilon}_t.$$

Proof. In the spirit of Eqn. (12) and Eqn. (4), let $\mathbf{a}_\sigma\left(t, \Delta t\right) \coloneqq \sqrt{\left(1-\bar{\alpha}_{t-\Delta t}-\sigma_t^2\right)/\left(1-\bar{\alpha}_t\right)}$ and $\mathbf{K}_t \coloneqq \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)$, then our reverse iterate is able to be rewritten as

$$q_\sigma\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t, \boldsymbol{\theta}_0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{a}_\sigma\left(t, 1\right)\left(\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0\right) + \mathbf{K}_{t-1} - \mathbf{a}_\sigma\left(t, 1\right)\mathbf{K}_t, \sigma_t^2\mathbf{I}\right). \tag{13}$$

Next, we show that it still satisfies the desired marginal distribution

$$q_\sigma\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \mathbf{K}_t, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right) = q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right).$$

The proof continues via backward induction on t. We can assume that the above equality holds for arbitrary $1 < t < T$. As the base case ($t = T$, isotropic Gaussian) has already been established,

[1] Please refer to Song et al. (2020b) for more details of "Variance-Exploding" SDE.

we then prove that $q_\sigma\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{K}_{t-1}, \left(1-\bar{\alpha}_{t-1}\right)\mathbf{I}\right)$, thereby completing the induction from T to 1. Using the Bayes' rule,

$$q_\sigma\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0\right) = q_\sigma\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0\right)\int_{\boldsymbol{\theta}_t} q_\sigma\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1},\boldsymbol{\theta}_0\right)d\boldsymbol{\theta}_t = \int_{\boldsymbol{\theta}_t} q_\sigma\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0\right)q_\sigma\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1},\boldsymbol{\theta}_0\right)d\boldsymbol{\theta}_t$$
$$= \int_{\boldsymbol{\theta}_t} q_\sigma\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)q_\sigma\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t,\boldsymbol{\theta}_0\right)d\boldsymbol{\theta}_t$$

From Lemma 2 ($\boldsymbol{\theta}_0$ is viewed as constants), we know that $q_\sigma\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0\right) = \mathcal{N}\left(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}\mathbf{I}\right)$ in which

$$\boldsymbol{\mu}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{a}_\sigma\left(t,1\right)\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \mathbf{K}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0\right) + \mathbf{K}_{t-1} - \mathbf{a}_\sigma\left(t,1\right)\mathbf{K}_t,$$
$$\boldsymbol{\Sigma}_{t-1}\mathbf{I} = \sigma_t^2\mathbf{I} + \mathbf{a}_\sigma^2\left(t,1\right)\left(1-\bar{\alpha}_t\right)\mathbf{I} = \sigma_t^2\mathbf{I} + \frac{1-\bar{\alpha}_{t-1}-\sigma_t^2}{1-\bar{\alpha}_t}\cdot\left(1-\bar{\alpha}_t\right)\mathbf{I}.$$

After canceling the highlighted terms, we have $q_\sigma\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{K}_{t-1}, \left(1-\bar{\alpha}_{t-1}\right)\mathbf{I}\right)$, which completes the induction step.

Then using the reparameterization trick, one can rewrite the DDIM-style iterate as

$$\boldsymbol{\theta}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{\theta}_0 + \mathbf{a}_\sigma\left(t,1\right)\left(\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0\right) + \mathbf{K}_{t-1} - \mathbf{a}_\sigma\left(t,1\right)\mathbf{K}_t + \sigma_t\boldsymbol{\epsilon}_t$$
$$\approx \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{\theta}}_0 + \kappa_{t-1}\mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t-1\right) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{\theta}}_0 - \kappa_t\mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t\right)}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t\boldsymbol{\epsilon}_t$$

from which the theorem conclusion follows. □

B.5.2 Adding Controllable Guidance

A good property of our proposal is the preserved ability to perform *conditional* sampling by decreasing the variance or range of noise inputs to the diffusion model at inference time. The intended effect is to decrease the diversity of the samples while increasing the quality of each individual sample.

In connection with score matching: Before we get started, it is noteworthy that prior work (i.e., KODM) adopts score-based generative modeling (Song & Ermon, 2019), where data are generated via Langevin dynamics (Welling & Teh, 2011) using score functions learned through score matching. Here, the score function of a distribution $p(\boldsymbol{\theta})$ is defined as its gradient $\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{\theta})$. By perturbing the data with Gaussian noise at multiple noise levels, we show below that learning **KDDM** is almost equivalent to learning these score functions.

Recall that $q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right), \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$ and therefore,

$$\nabla_{\boldsymbol{\theta}_t}\log q_\psi\left(\boldsymbol{\theta}_t\right) = \frac{\nabla_{\boldsymbol{\theta}_t} q_\psi\left(\boldsymbol{\theta}_t\right)}{q_\psi\left(\boldsymbol{\theta}_t\right)} = \frac{\int_{\boldsymbol{\theta}_0} q\left(\boldsymbol{\theta}_0\right)\nabla_{\boldsymbol{\theta}_t} q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)d\boldsymbol{\theta}_0}{q_\psi\left(\boldsymbol{\theta}_t\right)} \qquad \triangleright\ \nabla_x q(x) = q(x)\nabla_x\log q(x)$$
$$= \frac{1}{q_\psi\left(\boldsymbol{\theta}_t\right)}\int_{\boldsymbol{\theta}_0} q\left(\boldsymbol{\theta}_0\right)q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)\nabla_{\boldsymbol{\theta}_t}\log q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)d\boldsymbol{\theta}_0$$
$$= \int_{\boldsymbol{\theta}_0} q_\psi\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_t\right)\nabla_{\boldsymbol{\theta}_t}\log q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)d\boldsymbol{\theta}_0 = \mathbb{E}_{q(\boldsymbol{\theta}_0|\boldsymbol{\theta}_t)}[\nabla_{\boldsymbol{\theta}_t}\log q_\psi\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_0\right)]$$
$$= \mathbb{E}_{\boldsymbol{\theta}_0|\boldsymbol{\theta}_t}\left[\nabla_{\boldsymbol{\theta}_t}\left(-\frac{\left(\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 - \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)\right)^2}{2\left(1-\bar{\alpha}_t\right)}\right)\right]$$
$$= \mathbb{E}_{\boldsymbol{\theta}_0|\boldsymbol{\theta}_t}\left[\frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 - \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)}{1-\bar{\alpha}_t}\right] \approx -\frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{\theta}}_0 - \kappa_t\mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t\right)}{1-\bar{\alpha}_t} \tag{14}$$

Now, reorganizing the reverse sampling formulation from Eqn. (6), we can see that

$$\boldsymbol{\theta}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\theta_t - \kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)\right) + \kappa_{t-1}\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t-1\right) - \frac{\beta_t}{\sqrt{\alpha_t}}\nabla_{\boldsymbol{\theta}_t}\log q_\psi\left(\boldsymbol{\theta}_t\right) + \frac{\left(1-\bar{\alpha}_{t-1}\right)\beta_t}{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \tag{15}$$

This is structurally similar to time-reversed SDEs used in KODM (Song et al., 2026, Figure 1, page 2).

❶ **Training-Based Guidance.** To produce samples using **KDDM** from the conditional distribution $p(\boldsymbol{\theta}_0|\mathbf{c})$ for a desired condition $\mathbf{c}$ (e.g., one class associated with a high reward), a fundamental insight is to modify the unconditional score $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t)$ with the conditional score $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c})$ in Eqn. (15), by following the principle of *classifier guidance* (Dhariwal & Nichol, 2021). Based on the Bayes' rule, the new gradient term can be decomposed as

$$\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c}) = \nabla_{\boldsymbol{\theta}_t} \log \left(\frac{q_\psi(\boldsymbol{\theta}_t)\, q_\psi(\mathbf{c}|\boldsymbol{\theta}_t)}{\cancel{p(\mathbf{c})}} \right) = \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) + \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\mathbf{c}|\boldsymbol{\theta}_t) \tag{16}$$

So one is still able to steer the generation process by adding a pre-trained time-dependent classifier $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\mathbf{c}|\boldsymbol{\theta}_t)$. Specifically, with (tempered) guidance scale $\omega \geq 0$, we have the score function

$$\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c},\omega) \coloneqq \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) q_\psi(\mathbf{c}|\boldsymbol{\theta}_t)^\omega = \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) + \omega \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\mathbf{c}|\boldsymbol{\theta}_t)\,.$$

We formalize the class-conditional sampling procedure in Algorithm 3.

Algorithm 3 Classifier Guidance Sampling of **KDDM**

Require: Clean data predictor $\hat{\boldsymbol{\theta}}_0$, conditional label $\mathbf{c}$, classifier $p_\phi(\mathbf{c}|\boldsymbol{\theta}_t)$, and guidance scale ω

1: $\boldsymbol{\theta}_T \leftarrow$ sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, T-1, \ldots, 1$ **do**
3: Compute unconditional score $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) = -\frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{\theta}}_0 - \kappa_t \mathbf{K}_\psi(\hat{\boldsymbol{\theta}}_0, t)}{1-\bar{\alpha}_t}$ ▷ Eqn. (14)
4: $\boldsymbol{\theta}_{t-1} \leftarrow$ sample from $\mathcal{N}\left(\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t\right)$ through annealed Langevin dynamics ▷ Eqn. (15)
5: Add classifier gradient $\boldsymbol{\theta}_{t-1} = \boldsymbol{\theta}_{t-1} - \frac{\beta_t}{\sqrt{\bar{\alpha}_t}} \cdot \omega \nabla_{\boldsymbol{\theta}_t} \log p_\phi(\mathbf{c}|\boldsymbol{\theta}_t)$ ▷ Eqn. (16)
6: **end for**
7: **return** $\boldsymbol{\theta}_0$

Algorithm 4 Classifier-Free Guidance Sampling of **KDDM**

Require: Clean data (conditional) predictor $\hat{\boldsymbol{\theta}}_0$, conditional label $\mathbf{c}$, and guidance scale ω

1: $\boldsymbol{\theta}_T \leftarrow$ sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, T-1, \ldots, 1$ **do**
3: Predict unconditional score $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) = -\frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{\theta}}_0(\boldsymbol{\theta}_t, t, \varnothing) - \kappa_t \mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0(\boldsymbol{\theta}_t, t, \varnothing), t\right)}{1-\bar{\alpha}_t}$
4: Predict conditional score $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c}) = -\frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{\theta}}_0(\boldsymbol{\theta}_t, t, \mathbf{c}) - \kappa_t \mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0(\boldsymbol{\theta}_t, t, \mathbf{c}), t\right)}{1-\bar{\alpha}_t}$
5: $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c},\omega) = \omega \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c}) + (1-\omega) \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t)$ ▷ Eqn. (17)
6: $\boldsymbol{\theta}_{t-1} \leftarrow$ sample from $\mathcal{N}\left(\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t\right)$ through annealed Langevin dynamics ▷ Eqn. (15)
7: **end for**
8: **return** $\boldsymbol{\theta}_0$

By the similar logic, **KDDM** can be easily extended to *classifier-free guidance* (Ho & Salimans, 2022) that eliminates the need for a separate classifier, as

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c},\omega) &= \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) + \omega \left(\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c}) - \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) \right) \\ &= \omega \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t|\mathbf{c}) + (1-\omega) \nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) \end{aligned} \tag{17}$$

Here a single model is sufficient to learn both score functions by treating $\mathbf{c}$ as an additional input: the training procedure is unified by randomly replacing $\mathbf{c}$ with the null input $\varnothing$ with a fixed probability. The above process is depicted in Algorithm 4.

❷ **Training-Free Guidance.** Also notice our diffusion model may support conditional generation in a training-free manner (Ye et al., 2024). Despite variations in implementation and application, we achieve this in **KDDM** by approximating $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\mathbf{c}|\boldsymbol{\theta}_t)$ expressed in Eqn. (16) through "soft" alignment or update correction.

Algorithm 5 Replacement-based Imputation of **KDDM**

Require: Clean data predictor $\hat{\boldsymbol{\theta}}_0$, Observation Ω_0
1: $\boldsymbol{\theta}_T \leftarrow$ sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, T-1, \ldots, 1$ **do**
3: Predict unknown part $\bar{\Omega}_{0|t} = \bar{\Omega}\left(\hat{\boldsymbol{\theta}}_0(\boldsymbol{\theta}_t, t)\right)$
4: $\boldsymbol{\epsilon} \leftarrow$ sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Perturb known part $\hat{\Omega}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t \Omega\left(\mathbf{K}_\psi\left(\left[\bar{\Omega}_{0|t}; \Omega_0\right], t\right)\right) + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
6: Replace known part $\hat{\boldsymbol{\theta}}_t = \left[\bar{\Omega}_t, \hat{\Omega}_t\right]$ ▷ Eqn. (18)
7: Compute $\boldsymbol{\theta}_{t-1}$ using **KDDM** or **KDIM** update with $\hat{\boldsymbol{\theta}}_t$ ▷ Eqn. (6) or Eqn. (12)
8: **end for**
9: **return** $\boldsymbol{\theta}_0$

The first method is *replacement* (Lugmayr et al., 2022; Ho et al., 2022; Yuan & Qiao, 2024) for *imputation* tasks. Denote by $\Omega_t = \Omega(\boldsymbol{\theta}_t)$ and $\bar{\Omega}_t = \bar{\Omega}(\boldsymbol{\theta}_t)$ the known and unknown oscillators of $\boldsymbol{\theta}_t$ respectively, then $\boldsymbol{\theta}_t = \left[\bar{\Omega}_t; \Omega_t\right]$ where $[u; v]$ means denotes a pre-defined element-wise concatenation of u and v. Our goal is to sample from $p\left(\bar{\Omega}_0 \middle| \mathbf{c} = \Omega_0\right)$. This is in general intractable, because one need computing $\nabla_{\boldsymbol{\theta}_t} \log q_\psi\left(\boldsymbol{\theta}_t|\mathbf{c}\right)$ (here $\nabla_{\bar{\Omega}_t} \log q_\psi\left(\bar{\Omega}_t|\mathbf{c}\right)$ instead). But we have

$$\begin{aligned} q_\psi\left(\bar{\Omega}_t|\mathbf{c}\right) &= \int q_\psi\left(\bar{\Omega}_t, \Omega_t|\mathbf{c}\right) d\Omega_t = \int q_\psi\left(\Omega_t|\mathbf{c}\right) q_\psi\left(\bar{\Omega}_t|\Omega_t, \mathbf{c}\right) d\Omega_t \\ &= \mathbb{E}_{q_\psi(\Omega_t|\mathbf{c})}\left[q_\psi\left(\bar{\Omega}_t|\Omega_t, \mathbf{c}\right)\right] \approx \mathbb{E}_{q_\psi(\Omega_t|\mathbf{c})}\left[q_\psi\left(\bar{\Omega}_t|\Omega_t\right)\right]. \end{aligned}$$

Thus, by randomly sampling from $q_\psi\left(\Omega_t|\mathbf{c}\right)$, i.e., $\hat{\Omega}_t = \sqrt{\bar{\alpha}_t}\mathbf{c} + \kappa_t \Omega\left(\mathbf{K}_\psi\left(\tilde{\boldsymbol{\theta}}_0, t\right)\right) + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ in which $\tilde{\boldsymbol{\theta}}_0 \coloneqq \left[\bar{\Omega}\left(\hat{\boldsymbol{\theta}}_0\right); \Omega_0\right]$, we obtain the approximation:

$$\nabla_{\bar{\Omega}_t} \log q_\psi\left(\bar{\Omega}_t|\mathbf{c}\right) \simeq \nabla_{\bar{\Omega}_t} \log q_\psi\left(\bar{\Omega}_t \middle| \hat{\Omega}_t\right) = \nabla_{\bar{\Omega}_t} \log q_\psi\left(\left[\bar{\Omega}_t; \hat{\Omega}_t\right]\right) \tag{18}$$

The detailed sampling process is shown in Algorithm 5.

Algorithm 6 Posterior Guidance Sampling of **KDDM**

Require: Clean data predictor $\hat{\boldsymbol{\theta}}_0$, conditional label $\mathbf{c}$, and guidance strength $\{\zeta_t\}_{t=1}^T$
1: $\boldsymbol{\theta}_T \leftarrow$ sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, T-1, \ldots, 1$ **do**
3: Compute unconditional score $\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) = -\frac{\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{\theta}}_0 - \kappa_t \mathbf{K}_\psi\left(\hat{\boldsymbol{\theta}}_0, t\right)}{1-\bar{\alpha}_t}$ ▷ Eqn. (14)
4: $\boldsymbol{\theta}_{t-1} \leftarrow$ sample from $\mathcal{N}\left(\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t\right)$ through annealed Langevin dynamics ▷ Eqn. (15)
5: Predict posterior mean $\hat{\boldsymbol{\theta}}_{0|t} = \hat{\boldsymbol{\theta}}_0(\boldsymbol{\theta}_t, t)$ ▷ Eqn. (20)
6: Approximate gradient $\nabla_{\boldsymbol{\theta}_t} \log q_\psi\left(\mathbf{c}|\boldsymbol{\theta}_t\right) \approx \nabla_{\boldsymbol{\theta}_t} \textsc{Metrics}\left(\hat{\boldsymbol{\theta}}_{0|t}, \mathbf{c}\right)$ ▷ Eqn. (21)
7: Add posterior guidance $\boldsymbol{\theta}_{t-1} = \boldsymbol{\theta}_{t-1} - \zeta_t \nabla_{\boldsymbol{\theta}_t} \log p_\phi(\mathbf{c}|\boldsymbol{\theta}_t)$ ▷ Eqn. (16)
8: **end for**
9: **return** $\boldsymbol{\theta}_0$

The second method is *posterior sampling* (Li et al., 2022; Chung et al., 2023; Song et al., 2023). We consider an independent graphical model: $\boldsymbol{\theta}_0 \to \mathbf{c}$ and $\boldsymbol{\theta}_0 \to \boldsymbol{\theta}_t$. Then factorizing $q_\psi\left(\mathbf{c}|\boldsymbol{\theta}_t\right)$ in Eqn. (16) yields the following relationship:

$$q_\psi\left(\mathbf{c}|\boldsymbol{\theta}_t\right) = \int_{\boldsymbol{\theta}_0} q_\psi\left(\mathbf{c}|\boldsymbol{\theta}_0, \boldsymbol{\theta}_t\right) q_\psi\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_t\right) d\boldsymbol{\theta}_0 = \int_{\boldsymbol{\theta}_0} q\left(\mathbf{c}|\boldsymbol{\theta}_0\right) q_\psi\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_t\right) d\boldsymbol{\theta}_0 \tag{19}$$

The right equality stems from the Markov property of forward process. Now the problem is transformed into calculating the intractable $q_\psi\left(\boldsymbol{\theta}_0|\boldsymbol{\theta}_t\right)$ because the likelihood of $q\left(\mathbf{c}|\boldsymbol{\theta}_0\right)$ is generally

deterministic. Fortunately, we are able to approximate $q_\psi(\boldsymbol{\theta}_0|\boldsymbol{\theta}_t)$ with its posterior mean. The core idea is to use Tweedie's Formula (see Lemma 8). Rewrite $\mathbf{x}$ as $\boldsymbol{\theta}_t$ and $\boldsymbol{\eta}$ as $\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0 + \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t)$, and note that in KDDM, $\mathbf{x}|\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{\eta}, (1-\bar{\alpha}_t)\mathbf{I})$ and $p_0(\mathbf{x}) = \mathcal{N}(\mathbf{0}, (1-\bar{\alpha}_t)\mathbf{I})$. Therefore,

$$p(\mathbf{x}|\boldsymbol{\eta}) = \frac{1}{(2\pi(1-\bar{\alpha}_t))^{d/2}} \exp\left(-\frac{\|\mathbf{x}\|^2}{2(1-\bar{\alpha}_t)}\right) \exp\left(\frac{\boldsymbol{\eta}^\top \mathbf{x}}{1-\bar{\alpha}_t} - \frac{\|\boldsymbol{\eta}\|^2}{2(1-\bar{\alpha}_t)}\right).$$

By setting $F(\mathbf{x}) = \mathbf{x}/(1-\bar{\alpha}_t)$ and $L(\boldsymbol{\eta}) = \|\boldsymbol{\eta}\|^2/(2(1-\bar{\alpha}_t))$, we obtain

$$\hat{\boldsymbol{\eta}} = \frac{\nabla_{\mathbf{x}} \log q_\psi(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_0(\mathbf{x})}{(\nabla_{\mathbf{x}} F(\mathbf{x}))^\top} = \frac{\nabla_{\mathbf{x}} \log q_\psi(\mathbf{x}) + \frac{\mathbf{x}}{(1-\bar{\alpha}_t)}}{\frac{1}{(1-\bar{\alpha}_t)}} = \mathbf{x} + (1-\bar{\alpha}_t)\nabla_{\mathbf{x}} \log q_\psi(\mathbf{x}).$$

As such, based on Eqn. (14), the unique posterior mean of $\boldsymbol{\theta}_0$ given $\boldsymbol{\theta}_t$ is

$$\hat{\boldsymbol{\theta}}_{0|t} = \frac{1}{\sqrt{\bar{\alpha}_t}}(\hat{\boldsymbol{\eta}} - \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t)) = \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{\theta}_t + (1-\bar{\alpha}_t)\nabla_{\boldsymbol{\theta}_t} \log q_\psi(\boldsymbol{\theta}_t) - \kappa_t \mathbf{K}_\psi(\boldsymbol{\theta}_0, t)) \approx \hat{\boldsymbol{\theta}}_0 \tag{20}$$

With Eqn. (19) and the mapping: $\boldsymbol{\theta}_t \to \mathbb{E}[\boldsymbol{\theta}_0|\boldsymbol{\theta}_t] = \hat{\boldsymbol{\theta}}_{0|t}$, we can define a new approximation as

$$\nabla_{\boldsymbol{\theta}_t} q_\psi(\mathbf{c}|\boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}_t} \int_{\boldsymbol{\theta}_0} q(\mathbf{c}|\boldsymbol{\theta}_0)\, q_\psi(\boldsymbol{\theta}_0|\boldsymbol{\theta}_t)\, d\boldsymbol{\theta}_0 = \nabla_{\boldsymbol{\theta}_t} \mathbb{E}_{\boldsymbol{\theta}_0|\boldsymbol{\theta}_t}[q(\mathbf{c}|\boldsymbol{\theta}_0)] \simeq \nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right) \tag{21}$$

Regarding the error bound of the posterior approximation, we provide it in Theorem 3 as follows.

Theorem 3. *Assume that (i) the posterior satisfies $\boldsymbol{\theta}_0|\boldsymbol{\theta}_t \sim \mathcal{N}(\hat{\boldsymbol{\theta}}_{0|t}, \boldsymbol{\Sigma})$; (ii) $\hat{\boldsymbol{\theta}}_{0|t}$ is differentiable w.r.t. $\boldsymbol{\theta}_t$; (iii) the density $q(\mathbf{c}|\boldsymbol{\theta}_t)$ has bounded third derivative: $\sup_{\mathbf{x}} \|\mathrm{D}^3 q(\mathbf{c}|\mathbf{x})\| \leq \mathbf{M}_3$. Then*

$$\left\|\nabla_{\boldsymbol{\theta}_t} q_\psi(\mathbf{c}|\boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right)\right\| \leq \frac{1}{2}\left\|\frac{\partial \hat{\boldsymbol{\theta}}_{0|t}}{\partial \boldsymbol{\theta}_t}\right\| \mathbf{M}_3 \operatorname{tr}(\boldsymbol{\Sigma}).$$

Proof. Let $\mathbf{J}_{\hat{\boldsymbol{\theta}}} \coloneqq \frac{\partial \hat{\boldsymbol{\theta}}_{0|t}}{\partial \boldsymbol{\theta}_t}$, $\varphi(\boldsymbol{\theta}_t) \coloneqq \mathbb{E}_{\boldsymbol{\theta}_0|\boldsymbol{\theta}_t}[q(\mathbf{c}|\boldsymbol{\theta}_0)]$ and $\phi(\boldsymbol{\theta}_t) \coloneqq q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right)$. Then based on Eqn. (21), we know that

$$\left\|\nabla_{\boldsymbol{\theta}_t} q_\psi(\mathbf{c}|\boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right)\right\| = \|\nabla_{\boldsymbol{\theta}_t} \varphi(\boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}_t} \phi(\boldsymbol{\theta}_t)\| \tag{22}$$

Using reparameterization trick, since $\boldsymbol{\theta}_0|\boldsymbol{\theta}_t \sim \mathcal{N}\left(\hat{\boldsymbol{\theta}}_{0|t}, \boldsymbol{\Sigma}\right)$, we can write $\boldsymbol{\theta}_0|\boldsymbol{\theta}_t = \hat{\boldsymbol{\theta}}_{0|t} + \varepsilon$, where $\varepsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. For the calculation of $\nabla_{\boldsymbol{\theta}_t} \varphi(\boldsymbol{\theta}_t)$, we can write

$$\nabla_{\boldsymbol{\theta}_t} \varphi(\boldsymbol{\theta}_t) = \mathbf{J}_{\hat{\boldsymbol{\theta}}}^\top \nabla_{\boldsymbol{\theta}_t} \mathbb{E}_\varepsilon\left[q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t} + \varepsilon\right)\right] = \mathbf{J}_{\hat{\boldsymbol{\theta}}}^\top \mathbb{E}_\varepsilon\left[\nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t} + \varepsilon\right)\right],$$

and for $\nabla_{\boldsymbol{\theta}_t} \phi(\boldsymbol{\theta}_t)$, we have $\nabla_{\boldsymbol{\theta}_t} \phi(\boldsymbol{\theta}_t) = \mathbf{J}_{\hat{\boldsymbol{\theta}}}^\top \nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right)$. Putting them together,

$$\nabla_{\boldsymbol{\theta}_t} \varphi(\boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}_t} \phi(\boldsymbol{\theta}_t) = \mathbf{J}_{\hat{\boldsymbol{\theta}}}^\top \left(\mathbb{E}_\varepsilon\left[\nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t} + \varepsilon\right)\right] - \nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right)\right) \tag{23}$$

In what follows, we focus on the bound of $\mathbb{E}_\varepsilon\left[\nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t} + \varepsilon\right)\right] - \nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right)$. By denoting $\nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t} + \varepsilon\right)$ as $f(\mathbf{x})$ in which $\mathbf{x} \coloneqq \hat{\boldsymbol{\theta}}_{0|t} + \varepsilon$, we first approximate f using a Taylor expansion around $\mathbf{x} = \hat{\boldsymbol{\theta}}_{0|t}$ as: $f(\mathbf{x}) = \nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right) + \mathbf{H}_f(\boldsymbol{\theta}_{0|t})\varepsilon + \mathbf{R}(\varepsilon)$, where $\mathbf{H}_f$ and $\mathbf{R}$ are Hessian and Taylor remainder respectively. Note that $\mathbb{E}[\varepsilon] = \mathbf{0}$, this gives

$$\mathbb{E}_\varepsilon\left[\nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t} + \varepsilon\right)\right] - \nabla_{\boldsymbol{\theta}_t} q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t}\right) = \mathbb{E}_\varepsilon[\mathbf{H}_f(\boldsymbol{\theta}_{0|t})\varepsilon] + \mathbb{E}_\varepsilon[\mathbf{R}(\varepsilon)] = \mathbb{E}_\varepsilon[\mathbf{R}(\varepsilon)] \tag{24}$$

Also note that $\|\mathrm{D}^3 f(\mathbf{x})\| \leq \sup_{\mathbf{x}} \|\mathrm{D}^3 f(\mathbf{x})\| = \mathbf{M}_3$ and $\mathbb{E}[\varepsilon^\top \varepsilon] = \operatorname{tr}(\boldsymbol{\Sigma})$, we have

$$\|\mathbb{E}_\varepsilon[\mathbf{R}(\varepsilon)]\| = \left\|\mathbb{E}_\varepsilon\left[\frac{1}{2!}\mathrm{D}^3 q\left(\mathbf{c}\middle|\hat{\boldsymbol{\theta}}_{0|t} + \xi\boldsymbol{\theta}\right)[\varepsilon, \varepsilon]\right]\right\| \leq \frac{1}{2}\mathbf{M}_3 \mathbb{E}_\varepsilon[\|\varepsilon\|^2] = \frac{1}{2}\mathbf{M}_3 \operatorname{tr}(\boldsymbol{\Sigma}) \tag{25}$$

where $0 < \xi < 1$. Now combining Eqn. (23) to Ineqn. (25) yields the error bound

$$\|\nabla_{\boldsymbol{\theta}_t}\varphi(\boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}_t}\phi(\boldsymbol{\theta}_t)\| = \left\|\mathbf{J}_{\hat{\boldsymbol{\theta}}}^{\top}\mathbb{E}_{\varepsilon}[\mathbf{R}(\varepsilon)]\right\| \leq \frac{1}{2}\left\|\mathbf{J}_{\hat{\boldsymbol{\theta}}}\right\|\mathbf{M}_3\mathrm{tr}(\boldsymbol{\Sigma}),$$

which finishes the proof with Eqn. (22). □

Incorporation of Eqn. (19) and Eqn. (21) into the usual ancestral sampling steps of **KDDM** leads to Algorithm 6, in which we construct our diffusion model in order to perform sampling from the posterior distribution.

❸ **Sequential Monte Carlo.** At last, we generalize **KDDM** to Monte Carlo sampling (Trippe et al., 2023; Cardoso et al., 2024; Lu et al., 2025b) from selections (or particles) with diffusion priors. At its core, the sequential Monte Carlo approach is to circumvent the intractability of $p(\boldsymbol{\theta}_0|\mathbf{c})$ by constructing a sequence of approximations with a bunch of weighted atoms. Specifically, we break the reverse process of **KDDM** into three stages for all $t \in [1, T]$:

S1 *Draw* $\boldsymbol{\theta}_{t-1}^{(1:N)} \sim \mathcal{N}\left(\cdot\middle|\boldsymbol{\theta}_t^{(1:N)}\right)$. Using Eqn. (6) or Eqn. (12), N particles are first sampled from an unconditional model as candidates, given $\boldsymbol{\theta}_t^{(1:N)}$ which are isotropic Gaussians when $t = T$.

S2 *Compute weights* $\omega_t^{(1:N)}$. These different candidates are then evaluated using either a learned classifier (as in ❶) or an implicit classifier (as in ❷). For instance, in imputation tasks, ω_t can be defined as the discrepancy from the observed portion, whereas in exploration tasks, it can be defined as the number of steps required to reach the goal.

S3 *Perform resampling.* After normalizing $\omega_t^{(1:N)}$, resampling steps (also known as particle filtering or branching mechanisms, Smith, 2013) discard particles with negligible weights and replicate those with larger weights. The resulting particles form the new set $\boldsymbol{\theta}_{t-1}^{(1:N)}$, after which the procedure returns to S1.

Algorithm 7 Sequential Monte Carlo Sampling of **KDDM**

Require: Clean data predictor $\hat{\boldsymbol{\theta}}_0$, evaluator $\boldsymbol{f}$, and conditional label $\mathbf{c}$

1: $\boldsymbol{\theta}_T^{(1:N)} \leftarrow$ sample N random candidates from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, T-1, \ldots, 1$ **do**
3: Compute $\boldsymbol{\theta}_{t-1}^{(1:N)}$ using **KDDM** or **KDIM** update with $\hat{\boldsymbol{\theta}}_t^{(1:N)}$ ▷ Eqn. (6) or Eqn. (12)
4: Re-weight candidates $\omega_t^{(1:N)} = \boldsymbol{f}\left(\boldsymbol{\theta}_{t-1}^{(1:N)}, \mathbf{c}\right)$ ▷ Using ❶ or ❷
5: Normalize sampling weights $\forall i,\ \tilde{\omega}_t^{(i)} \leftarrow \omega_t^{(i)} / \sum_{j=1}^{N} \omega_t^{(j)}$
6: Re-sample new candidates $\boldsymbol{\theta}_{t-1}^{(1:N)} \leftarrow \textsc{Resample}\left(\tilde{\omega}_t^{(1:N)}, \boldsymbol{\theta}_{t-1}^{(1:N)}\right)$ ▷ Algorithm 8
7: **end for**
8: **return** $\boldsymbol{\theta}_0$

Algorithm 7 summarizes these three steps: S1 (line 3), S2 (line 4), and S3 (line 5-6). It is noteworthy that for the choice of RESAMPLE function (line 6), we use residual resampling (Johansen, 2009) to mitigate the collapse of the sequential approximations into point masses, as presented in Algorithm 8. A main advantage of such a statistical approach is that it does not rely on any task-specific hyperparameters such as guidance strength for inference. In theory, the empirical distribution of the samples produced by **KDDM** converges to the target posterior when the number of particles goes to infinity. This almost sure convergence follows from standard asymptotics for particle filters; see Lemma 4 for details.

Algorithm 8 Residual Resample

Require: Weights $\omega_{1:N}$, Candidates $\mathbf{x}_{1:N}$

1: $(c_i, r_i) \leftarrow (\lfloor Nw_i \rfloor,\ Nw_i - \lfloor Nw_i \rfloor)$
2: $R = N - \sum_{i=1}^{N} c_i$
3: $\tilde{\mathbf{x}}_C \leftarrow [\underbrace{\mathbf{x}_1, \ldots, \mathbf{x}_1}_{c_1}, \ldots, \underbrace{\mathbf{x}_N, \ldots, \mathbf{x}_N}_{c_N}]$
4: $[i_1, \ldots, i_R] \sim \textsc{Multinomial}(r_{1:N}, R)$
5: **return** $\mathrm{concat}(\tilde{\mathbf{x}}_C, [\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_R}])$

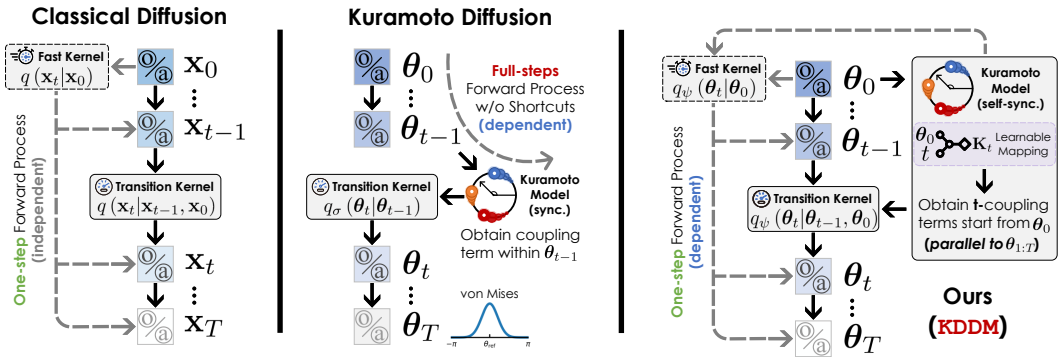


Figure 11: Comparison of the forward processes of different diffusion models.

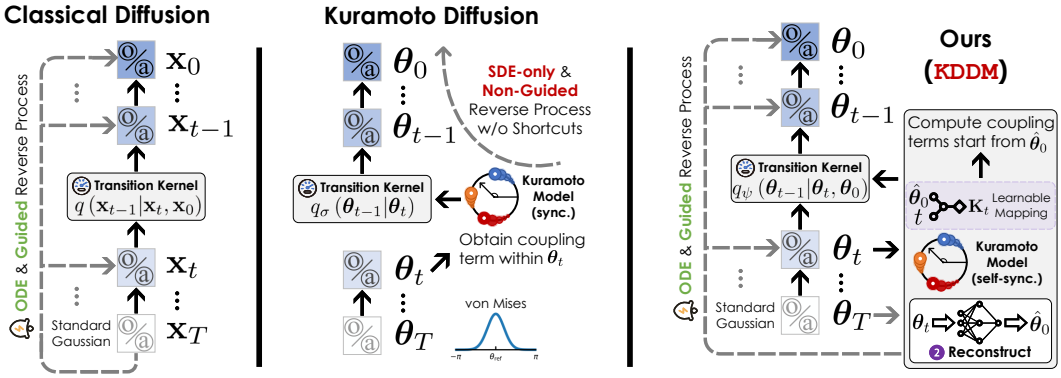


Figure 12: Comparison of the reverse processes of different diffusion models.

B.6 Understanding Why the Proposal Excels

In this subsection, we elaborate on the advantages of our method through a detailed comparison with DDPM and KODM. We compare the differences between the forward processes and reverse processes in Figs. 11 and 12, respectively.

In comparison with KODM. The main superiority of KDDM stems from how it handles the non-linear coupling in the Kuramoto model. Compared with KODM's efficiency limitations, KDDM offers two key advantages. First, we can directly derive a closed-form expression for the marginal distribution, which we refer to as one-step synchronization, substantially improving interpretability. Second, we can even skip intermediate steps and perform reverse desynchronization directly, enabling DDIM-style ODE acceleration. This greatly enhances the flexibility of our method, making the generation process both controllable and convenient. Since the above has been discussed in the main text, we now focus on the expected performance differences arising from the respective training and modeling approaches, as formulated in Eqns. (3) and (9). Building on Eqn. (14), this is essentially a comparison between *Global Score Matching* and *Local Score Matching*. Here, compared to directly learning the global score, the limitations of local score matching can be summarized as follows: (i) It only utilizes local transition information, making it difficult to directly capture the global probability geometry; (ii) Theoretically, it requires taking the expectation with respect to the unknown posterior $p(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t)$, which in practice can often only be approximated; (iii) Gradient estimation relies on specific predecessor states, leading to high variance

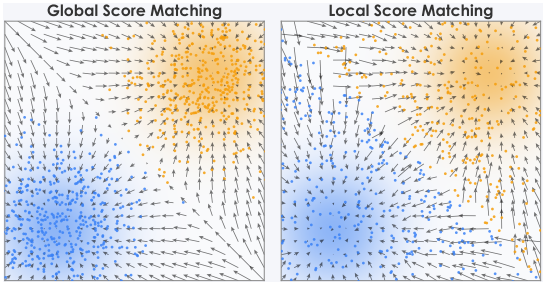


Figure 13: **Visualisations on the GMM.** From left to right, we plot the score field and samples (after 100 steps) of global SM and local SM respectively.

(see Fig. 13). Local errors are prone to accumulate gradually during sampling. Formally, we have the following proposition that highlights the sub-optimality of LSM in approximating our SM.

Proposition 2 (Sub-Optimality of LSM). *Let* $\boldsymbol{\mu} \coloneqq \mathbb{E}\left[\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right]$, $\boldsymbol{\Sigma} \coloneqq \mathrm{Cov}\left(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t\right)$, *and define* $g(\boldsymbol{\theta}_{t-1}) \coloneqq \nabla_{\boldsymbol{\theta}_t} \log p(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1})$, $\mathbf{J} \coloneqq \left.\frac{\partial g(x)}{\partial x}\right|_{x=\boldsymbol{\mu}}$. *Assume that* $g(\cdot)$ *is three-times differentiable around* $\boldsymbol{\theta}_{t-1} = \boldsymbol{\mu}$. *Then,*

$$\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\theta}_t} \log p(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}) - \nabla_{\boldsymbol{\theta}_t} \log p(\boldsymbol{\theta}_t)\right\|^2\right] \geq \sigma_{\min}(\mathbf{J})^2 \operatorname{tr}(\boldsymbol{\Sigma}) + \mathcal{O}\left(\|\boldsymbol{\Sigma}\|^{3/2}\right).$$

Proof. We start out by establishing the relationship between the global and local score functions. Applying the Fisher identity, i.e., $\nabla_x \log p(x) = \mathbb{E}_{z|x}[\nabla_x \log p\left(x|z\right)]$ whose related derivations can be found in Eqn. (14) or (Song et al., 2026, Section A.2, page 22), we obtain that

$$\nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t\right) = \mathbb{E}_{\boldsymbol{\theta}_{t-1} \sim p(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_t)}[\nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}\right)].$$

Note that $g\left(\boldsymbol{\theta}_{t-1}\right) = \nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}\right)$ and let $\delta \coloneqq \boldsymbol{\theta}_{t-1} - \boldsymbol{\mu}$, which means that $\nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}\right) = \mathbb{E}\left[g\left(\boldsymbol{\theta}_{t-1}\right)\right]$ and $\delta \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Sigma}\right)$. For g with the Taylor expansion on the point $\boldsymbol{\mu}$, we observe that $g\left(\boldsymbol{\theta}_{t-1}\right) = g\left(\boldsymbol{\mu}\right) + \mathbf{J}\delta + \frac{1}{2}\mathbf{H}\left[\delta, \delta\right] + \mathbf{R}\left(\boldsymbol{\theta}_{t-1}\right)$ where $\mathbf{H}$ is Hessian and $\mathbf{R}\left(\boldsymbol{\theta}_{t-1}\right) = \mathcal{O}\left(\delta^3\right)$. Then, as $\mathbb{E}\left[\delta\right] = \mathbf{0}$, we obtain that

$$\nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t\right) = \mathbb{E}\left[g\left(\boldsymbol{\mu}\right) + \mathbf{J}\delta + \frac{1}{2}\mathbf{H}\left[\delta, \delta\right] + \mathbf{R}\left(\boldsymbol{\theta}_{t-1}\right)\right] = g\left(\boldsymbol{\mu}\right) + \frac{1}{2}\mathbb{E}\left[\mathbf{H}\left[\delta, \delta\right]\right] + \mathbb{E}\left[\mathbf{R}\left(\boldsymbol{\theta}_{t-1}\right)\right]$$

As a result,

$$g\left(\boldsymbol{\theta}_{t-1}\right) - \nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t\right) = \mathbf{J}\delta + \frac{1}{2}\left(\mathbf{H}\left[\delta, \delta\right] - \mathbb{E}\left[\mathbf{H}\left[\delta, \delta\right]\right]\right) + \mathbf{R}\left(\boldsymbol{\theta}_{t-1}\right) - \mathbb{E}\left[\mathbf{R}\left(\boldsymbol{\theta}_{t-1}\right)\right].$$

To be precise, we define $\mathbf{Q} \coloneqq \frac{1}{2}\left(\mathbf{H}\left[\delta, \delta\right] - \mathbb{E}\left[\mathbf{H}\left[\delta, \delta\right]\right]\right)$ and $\tilde{\mathbf{R}} \coloneqq \mathbf{R}\left(\boldsymbol{\theta}_{t-1}\right) - \mathbb{E}\left[\mathbf{R}\left(\boldsymbol{\theta}_{t-1}\right)\right]$. With $\|\mathbf{J}\delta\|^2 = \mathcal{O}\left(\delta^2\right)$, $\mathbf{Q} = \mathcal{O}\left(\delta^2\right)$ and $\left(\mathbf{J}\delta\right)^\top \mathbf{Q} = \mathcal{O}\left(\delta^3\right)$ thus gives that

$$\begin{aligned} &\left\|g\left(\boldsymbol{\theta}_{t-1}\right) - \nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t\right)\right\|^2 = \|\mathbf{J}\delta\|^2 + 2\left(\mathbf{J}\delta\right)^\top \mathbf{Q} + \|\mathbf{Q}\|^2 + \mathcal{O}\left(\|\delta\|^5\right) \\ &\Rightarrow \mathbb{E}\left[\left\|g\left(\boldsymbol{\theta}_{t-1}\right) - \nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t\right)\right\|^2\right] = \mathbb{E}\left[\|\mathbf{J}\delta\|^2\right] + \mathcal{O}\left(\|\boldsymbol{\Sigma}\|^{3/2}\right) \end{aligned} \tag{26}$$

in which the second line is from $\mathbb{E}\left[\left\|\delta^{(\cdot)}\right\|\right] = \mathcal{O}\left(\|\boldsymbol{\Sigma}\|^{(\cdot)/2}\right)$. Since the underlying distribution is Gaussian, we have $\mathbb{E}\left[\|\mathbf{J}\delta\|^2\right] = \operatorname{tr}\left(\mathbf{J}\boldsymbol{\Sigma}\mathbf{J}^\top\right)$. Now using the definition of the smallest singular value $\sigma_{\min}(\mathbf{J}) \coloneqq \min_{\|x\|=1} \|\mathbf{J}x\|$, we have that for any vector v, $v^\top \mathbf{J}\boldsymbol{\Sigma}\mathbf{J}v = \left(\mathbf{J}^\top v\right)^\top \boldsymbol{\Sigma}\left(\mathbf{J}^\top v\right)$. Letting $y = \mathbf{J}^\top v$ and using the inequality $\|\mathbf{J}^\top v\| \geq \sigma_{\min}(\mathbf{J})\|v\|$, we thus have $\mathbf{J}\boldsymbol{\Sigma}\mathbf{J}^\top \succeq \sigma_{\min}(\mathbf{J})^2\boldsymbol{\Sigma}$.

Combining all the above, we deal with Eqn. (26):

$$\mathbb{E}\left[\left\|g\left(\boldsymbol{\theta}_{t-1}\right) - \nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t\right)\right\|^2\right] = \operatorname{tr}\left(\mathbf{J}\boldsymbol{\Sigma}\mathbf{J}^\top\right) + \mathcal{O}\left(\|\boldsymbol{\Sigma}\|^{3/2}\right) \geq \sigma_{\min}(\mathbf{J})^2 \operatorname{tr}\left(\boldsymbol{\Sigma}\right) + \mathcal{O}\left(\|\boldsymbol{\Sigma}\|^{3/2}\right).$$

Notice that $g\left(\boldsymbol{\theta}_{t-1}\right) = \nabla_{\boldsymbol{\theta}_t} \log p\left(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}\right)$, which completes the proof. □

Therefore, unless $\boldsymbol{\Sigma} = \mathbf{0}$ (degenerate posterior) or $\mathbf{J} = \mathbf{0}$, the local score cannot coincide with the global score, implying an intrinsic sub-optimality of local score matching.

In comparison with DDPM. We can clearly see that **KDDM** introduces synchronization effects on top of DDPM, thereby incorporating spatio-temporal information, which is important for coherent decision-making. **KDDM** retains almost all good properties of DDPM, and in fact, removing the synchronization term reduces **KDDM** exactly to DDPM. More intriguingly, beyond the structural improvements to the data, we find that incorporating the Kuramoto model may also yield gains in distribution modeling: if the fitting objective converges to its optimum, i.e., a perfect reconstructor, **KDDM** might achieve a higher log-likelihood.

In what follows, we verify this with a toy decision making example. To be more specific, we consider the ground-truth action sequences or trajectories with internal relationship as $\boldsymbol{\theta}_0 \sim \mathcal{N}\left(\mu_\psi, \sigma_\psi^2 \mathbf{I}\right)$. We aim to calculate and compare the *optimal* estimation error at diffusion step t in DDPM and in **KDDM**: $\mathbb{E}\left[\|e\|_2^2\right] := \mathbb{E}\left[\|\boldsymbol{\theta}_0 - \mathbb{E}\left[\boldsymbol{\theta}_0|\boldsymbol{\theta}_t\right]\|_2^2\right]$, which is the lower the better.

For DDPM, since $\boldsymbol{\theta}_t|\boldsymbol{\theta}_0 \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\theta}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$, we have $\boldsymbol{\theta}_t \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mu_\psi, \left(\bar{\alpha}_t\sigma_\psi^2 + 1 - \bar{\alpha}_t\right)\mathbf{I}\right)$, and $\operatorname{Cov}\left(\boldsymbol{\theta}_0, \boldsymbol{\theta}_t\right) = \mathbb{E}\left[\left(\boldsymbol{\theta}_0 - \mu_\psi\right)\left(\sqrt{\bar{\alpha}_t}\left(\boldsymbol{\theta}_0 - \mu_\psi\right) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}'\right)^\top\right]$ where $\boldsymbol{\epsilon}'$ is independently sampled from a standard Gaussian. Note that $\boldsymbol{\epsilon}' \perp \boldsymbol{\theta}_0$ and so,

$$\operatorname{Cov}\left(\boldsymbol{\theta}_0, \boldsymbol{\theta}_t\right) = \sqrt{\bar{\alpha}_t}\operatorname{Var}\left(\boldsymbol{\theta}_0\right) + \sqrt{1-\bar{\alpha}_t}\operatorname{Cov}\left(\boldsymbol{\theta}_0, \boldsymbol{\epsilon}'\right) = \sqrt{\bar{\alpha}_t}\sigma_\psi^2\mathbf{I}.$$

According to Lemma 3 and parameterizing $\boldsymbol{\theta}_0 = \mu_\psi + \sigma_\psi\boldsymbol{\epsilon}''$, $\boldsymbol{\epsilon}'' \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$, its optimal estimation error can be computed as

$$\begin{aligned}
\mathbb{E}\left[\|e\|_2^2\right] &= \mathbb{E}\left[\left\|\boldsymbol{\theta}_0 - \left(\mu_\psi + \operatorname{Cov}\left(\boldsymbol{\theta}_0, \boldsymbol{\theta}_t\right)\operatorname{Var}\left(\boldsymbol{\theta}_t\right)^{-1}\left(\boldsymbol{\theta}_t - \sqrt{\bar{\alpha}_t}\mu_\psi\right)\right)\right\|_2^2\right] \\
&= \mathbb{E}\left[\left\|\boldsymbol{\theta}_0 - \left(\mu_\psi - \frac{\sqrt{\bar{\alpha}_t}\sigma_\psi^2}{\bar{\alpha}_t\sigma_\psi^2 + 1 - \bar{\alpha}_t}\left(\sqrt{\bar{\alpha}_t}\mu_\psi - \boldsymbol{\theta}_t\right)\right)\right\|_2^2\right] \\
&= \mathbb{E}\left[\left\|\sigma_\psi\boldsymbol{\epsilon}'' - \frac{\sqrt{\bar{\alpha}_t}\sigma_\psi^2}{\bar{\alpha}_t\sigma_\psi^2 + 1 - \bar{\alpha}_t}\left(\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}' + \sqrt{\bar{\alpha}_t}\sigma_\psi\boldsymbol{\epsilon}''\right)\right\|_2^2\right] \\
&= \mathbb{E}\left[\left\|\frac{\sigma_\psi\left(1-\bar{\alpha}_t\right)}{\bar{\alpha}_t\sigma_\psi^2 + 1 - \bar{\alpha}_t}\boldsymbol{\epsilon}'' - \frac{\sqrt{\bar{\alpha}_t}\left(1-\bar{\alpha}_t\right)\sigma_\psi^2}{\bar{\alpha}_t\sigma_\psi^2 + 1 - \bar{\alpha}_t}\boldsymbol{\epsilon}'\right\|_2^2\right] \qquad \triangleright\ \mathbb{E}\left[\boldsymbol{\epsilon}''\boldsymbol{\epsilon}'^\top\right] = 0 \\
&= d \cdot \frac{\sigma_\psi^2\left(1-\bar{\alpha}_t\right)\left(\bar{\alpha}_t\sigma_\psi^2 + 1 - \bar{\alpha}_t\right)}{\left(\bar{\alpha}_t\sigma_\psi^2 + 1 - \bar{\alpha}_t\right)^2} = d \cdot \frac{\sigma_\psi^2\left(1-\bar{\alpha}_t\right)}{\bar{\alpha}_t\sigma_\psi^2 + 1 - \bar{\alpha}_t}
\end{aligned} \tag{27}$$

For **KDDM**, we assume that the value of a Kuramoto model $\kappa_t\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)$ can be approximated by a linear model $K_\psi(\boldsymbol{\theta}_0, t)\boldsymbol{\theta}_0$ in which $K_\psi(\boldsymbol{\theta}_0, t)$ involves coefficients of all oscillators at t, since here we are able to ignore the nonlinear intermediate processes here and focus solely on the outcome. Using Eqn. (5), this results in $\boldsymbol{\theta}_t|\boldsymbol{\theta}_0 \sim \mathcal{N}\left(\left(\sqrt{\bar{\alpha}_t} + K_\psi(\boldsymbol{\theta}_0, t)\right)\boldsymbol{\theta}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$. For simplicity, we denote $\sqrt{\bar{\alpha}_t} + K_\psi(\boldsymbol{\theta}_0, t)$ as K_t. Then, similar to the DDPM case, we can obtain that

$$\begin{aligned}
\mathbb{E}\left[\|e\|_2^2\right] &= \mathbb{E}\left[\left\|\boldsymbol{\theta}_0 - \left(\mu_\psi + \operatorname{Cov}\left(\boldsymbol{\theta}_0, \boldsymbol{\theta}_t\right)\operatorname{Var}\left(\boldsymbol{\theta}_t\right)^{-1}\left(\boldsymbol{\theta}_t - K_t\mu_\psi\right)\right)\right\|_2^2\right] \\
&= \mathbb{E}\left[\left\|\boldsymbol{\theta}_0 - \left(\mu_\psi - \frac{K_t\sigma_\psi^2}{K_t^2\sigma_\psi^2 + 1 - \bar{\alpha}_t}\left(\boldsymbol{\theta}_t - K_t\mu_\psi\right)\right)\right\|_2^2\right] \\
&= d \cdot \frac{\sigma_\psi^2\left(1-\bar{\alpha}_t\right)}{\left(\sqrt{\bar{\alpha}_t} + K_\psi(\boldsymbol{\theta}_0, t)\right)^2\sigma_\psi^2 + 1 - \bar{\alpha}_t}
\end{aligned} \tag{28}$$

Comparing Eqn. (27) and Eqn. (28), we can see that **KDDM** has theoretically lower error of the Bayes estimator than DDPM, given $\left(\sqrt{\bar{\alpha}_t} + K_\psi(\boldsymbol{\theta}_0, t)\right)^2 \geq \bar{\alpha}_t$ when the linear coefficient $K_\psi(\boldsymbol{\theta}_0, t)$ is non-negative.

C Experimental Setup

This section provides key details of our experimental setup to facilitate the reproducibility of our experiments. We first describe the environments and datasets used for trajectory generation and decision-making, followed by the baselines and evaluation metrics. We then introduce our physical testbed and provide a detailed account of the implementation of our method, including the training configurations and sampling procedures.

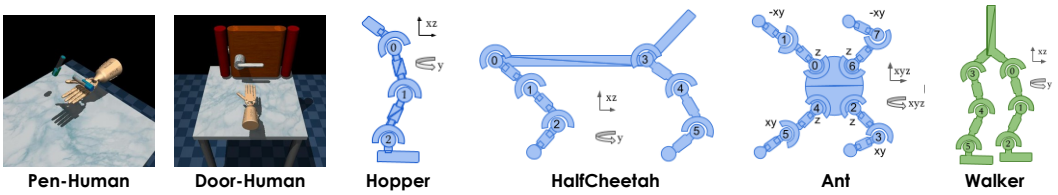


Figure 14: Gymnasium-Robotics (Towers et al., 2026) entities and datasets.

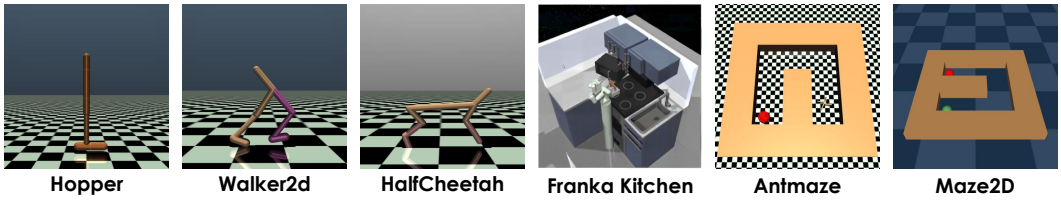


Figure 15: Offline reinforcement learning environments and datasets.

C.1 Environments and Datasets

C.1.1 Robotic Trajectory Generation

To evaluate the trajectory generation capabilities of diffusion models, we obtain six different datasets through the Minari API (Younis et al., 2024), from which these datasets are generated following Gymnasium-Robotics (Towers et al., 2026), a collection of robotics simulation environments for RL. The environments run with the MuJoCo physics engine and the maintained MuJoCo Python bindings. Fig. 14 visualizes the robotic embodiments and some tasks in the Adroit Hand environments.

Adroit Hand (Rajeswaran et al., 2017) consists of a Shadow Dexterous Hand attached to a free arm. There are 2 environments considered in our experiments: (i) Pen-Human (`pen-human-v2`) whose task to be completed consists of repositioning the blue pen to match the orientation of the green target, and (ii) Door-Human (`door-human-v2`) whose task to be completed consists of undoing the latch and swinging the door open. They are all 28-degree-of-freedom systems which consist of a 24-degree-of-freedom ShadowHand and a 4-degree-of-freedom arm.

MuJoCo (Peng et al., 2021) stands for Multi-Joint dynamics with physical contact between the robots and their environment. MuJoCo attempts at getting realistic physics simulations for possible physical contact dynamics by aiming for physical accuracy and computational efficiency. Also, four environments are considered here: (i) Hopper (`hopper-medium-v0`) whose goal is to make hops that move in the forward direction by applying torque to the three hinges that connect the four body parts; (ii) HalfCheetah (`halfcheetah-medium-v0`), a 2-dimensional robot consisting of 9 body parts and 8 joints connecting them, whose goal is to apply torque to the joints to make the cheetah run forward as fast as possible; (iii) Ant (`ant-medium-v0`), a 3D quadruped robot consisting of a torso (free rotational body) with four legs attached to it, whose goal is to coordinate the four legs to move in the forward direction by applying torque to the eight hinges connecting the two body parts of each leg and the torso; and (iv) Walker (`walker2d-medium-v0`), a two-dimensional bipedal robot consisting of seven main body parts, whose goal is to walk in the forward direction by applying torque to the six hinges connecting the seven body parts.

C.1.2 Offline Reinforcement Learning

We evaluate diffusion-based RL algorithms implemented with the `CleanDiffuser` library on the D4RL benchmark (Fu et al., 2020), one of the most widely used benchmarks for offline RL, including locomotion, manipulation, and navigation. Visualization of our tasks is presented in Fig. 15.

Gym-MuJoCo (Brockman et al., 2016) consists of three widely used offline RL locomotion tasks (HalfCheetah, Hopper, and Walker2d), where the goal is to control three MuJoCo robots to achieve high forward velocity while maintaining stable locomotion and minimizing energy consumption. For each task, D4RL provides three dataset qualities: *medium*, *medium-replay*, and *medium-expert*,

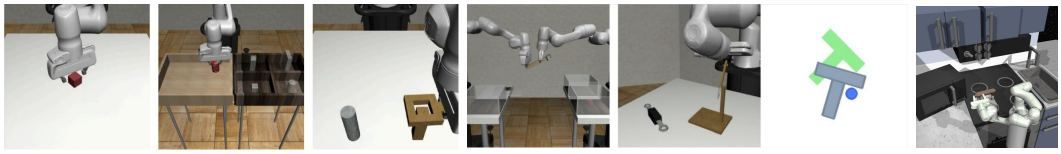


Figure 16: Offline imitation learning environments and datasets.

corresponding to medium-quality trajectories, all replay-buffer transitions collected before reaching medium performance, and an equal mixture of medium- and expert-quality trajectories, respectively.

Franka Kitchen (Gupta et al., 2019) requires controlling a realistic 9-DoF Franka robot arm to perform a series of household tasks in a kitchen environment. We consider two offline datasets, *partial* and *mixed*, both consisting of undirected demonstrations in which the robot performs subtasks that may not directly correspond to the target configuration. In the *partial* dataset, a subset of trajectories is guaranteed to contain successful task executions, allowing an imitation learning agent to succeed by identifying and exploiting the relevant data. In contrast, the *mixed* dataset contains no trajectory that completes the task in its entirety, requiring an RL agent to compose relevant sub-trajectories to accomplish the task. Consequently, *mixed* poses a substantially greater generalization challenge.

Maze (Fu et al., 2020) is a collection of environments in which an agent has to navigate through a maze to reach certain goal position. Two different agents can be used: a 2-DoF force-controlled ball (Maze2D), or the classic Ant agent (Antmaze). Specifically, Antmaze requires controlling an 8-DoF quadrupedal *Ant* robot to navigate through maze environments and reach designated endpoints. Meanwhile, the task in Maze2D is for a 2-DoF ball that is force-actuated in the Cartesian directions x and y to reach a target goal in a closed maze. In these offline datasets, the robot receives a reward only upon reaching the goal, while many trajectory segments fail to lead to the endpoint. This makes the task particularly challenging due to its sparse rewards and long-horizon decision-making. We evaluate performance using the success rate of reaching the endpoint, under which conventional offline RL algorithms often struggle to achieve competitive performance.

C.1.3 Offline Imitation Learning

We evaluate diffusion-based IL algorithms implemented with `CleanDiffuser` using both state- and image-based observations. Among these benchmarks, Relay Kitchen and Robomimic support both velocity and position control. For each algorithm, we select the action space that yields the best performance. Fig. 16 provides their visualizations, and further details are provided below:

PushT (Florence et al., 2022) requires pushing a T-shaped block toward a fixed target using a circular end-effector. The task involves complex, contact-rich object dynamics, requiring precise manipulation of the T-shaped block through point contacts. We consider three variants of the environment. The *PushT* environment uses a five-dimensional state space, consisting of the end-effector location (`agent_x`, `agent_y`) and the block's position and orientation (`block_x`, `block_y`, `block_angle`). The *PushT-keypoints* environment represents the block using nine 2D keypoints derived from its ground-truth pose, together with the end-effector location. The *PushT-Image* environment provides the end-effector location and a top-down RGB image as observations.

Relay Kitchen (Gupta et al., 2019) is commonly used to evaluate imitation learning capabilities. The environment consists of a 9-DoF position-controlled Franka robot arm interacting with a kitchen scene featuring an openable microwave, four rotatable oven burners, an oven light switch, a freely movable kettle, two hinged cabinets, and a sliding cabinet door. The *Relay* dataset contains 566 human demonstrations, each completing four tasks in an arbitrary order. The goal is to execute as many tasks as possible regardless of their order, thereby testing both short- and long-horizon multi-modal behavior.

Robomimic (Mandlekar et al., 2021) requires controlling a robot arm to perform complex manipulation tasks from a limited number of human demonstrations. Learning from human-collected datasets is particularly challenging due to the non-Markovian nature of human demonstrations and variations in demonstration quality, making them substantially more difficult than machine-generated datasets. The Proficient-Human (PH) and Multi-Human (MH) datasets are collected through remote teleoper-

ation. The PH datasets contain 200 demonstrations collected by a single experienced teleoperator, whereas the MH datasets contain 300 demonstrations collected by six teleoperators with varying levels of proficiency, each contributing 50 demonstrations. The benchmark comprises five PH tasks (*Lift*, *Can*, *Square*, *Tool_hang*, and *Transport*) and four MH tasks (*Lift*, *Can*, *Square*, and *Transport*). Each task provides both state- and image-based observations.

C.2 Baselines

C.2.1 Implemented Diffusion Models

DDPM/DDIM/DPM-Solver. DDPM (Ho et al., 2020), DDIM (Song et al., 2020a), DPM-Solver (Lu et al., 2022), and DPM-Solver++ (Lu et al., 2025a) share the same diffusion SDE/ODE formulation and noise schedule (Nichol & Dhariwal, 2021). Therefore, a single trained prediction model can support all four sampling schemes. For DDPM and DDIM, we directly implement their standard one-step updates. DPM-Solver instead solves the linear part of the diffusion ODE analytically and approximates the remaining nonlinear term with a Taylor expansion. Although higher-order solvers can reduce discretization error, we find them less stable under guided sampling. Therefore, we use DPM-Solver-1 by default for decision-making tasks. DPM-Solver++ improves stability by parameterizing the model to predict the clean data $\mathbf{x}_0$ rather than the noise.

EDM. EDM (Karras et al., 2022) parameterizes noisy samples as a scaled clean sample with additive noise, $\mathbf{x}_t = s_t(\mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} t)$. we follow the standard EDM setting with $s_t = 1$ and implements its reverse process using a data prediction model $D_\theta(\mathbf{x}; \sigma)$. The model adopts EDM's standard preconditioning, which combines a skip connection with a neural network F_θ and appropriately scales its input, output, and noise-level condition. During training, it samples noise levels, perturbs the data accordingly, and optimizes the weighted denoising objective prescribed by EDM. In implementation, EDM therefore requires only EDM preconditioning, noise-level conditioning, and the corresponding reverse sampler, while the backbone F_θ remains independent of these components.

Rectified Flow. Rectified Flow (Lipman et al., 2022) learns a velocity field that transports samples between two distributions along approximately straight trajectories. We construct intermediate states by linear interpolation between source and target samples and train the velocity field with a least-squares regression objective. The implementation supports arbitrary source and target distributions and integrates the learned ODE in either direction. By default, we use the dataset distribution as the source and a standard Gaussian distribution as the target, following the common generative setting. `CleanDiffuser` also supports R-Flow, which retrains the velocity field using generated samples to further straighten the trajectories and thereby reduce the number of sampling steps.

VE-SDE/VP-SDE. VE-SDE and VP-SDE (Song et al., 2020b) formulate the forward process as a continuous-time SDE. VE-SDE uses a variance-exploding SDE with geometric noise levels $\sigma_t = \sigma_{\min}(\sigma_{\max}/\sigma_{\min})^t$, so $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$ without rescaling the data; following `CleanDiffuser`, we parameterize the model as a denoiser $D_\theta(\mathbf{x}; \sigma) = \mathbf{x} + \sigma F_\theta(\mathbf{x}, \ln(\sigma/2))$ and weight the loss by $1/\sigma^2$. VP-SDE instead keeps the marginal variance bounded by a linear schedule β_t, with $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$, $\sigma_t = \sqrt{1 - \alpha_t^2}$, and the model predicts the noise conditioned on the time index. Both are sampled by an Euler–Maruyama integrator of the reverse SDE over 100 steps, and dropping the stochastic term yields the probability-flow ODE.

C.2.2 Diffusion Planners

Diffuser. Diffuser (Janner et al., 2022) formulates planning as conditional trajectory generation, a paradigm adopted by many subsequent diffusion-based planners. It models trajectory optimality through the cumulative reward and uses a reward-prediction network J_ϕ to guide the diffusion process. At each decision step, Diffuser fixes the current state as the first state of the trajectory and performs guided inpainting to generate candidate trajectories. It then selects the trajectory with the highest predicted cumulative reward and executes its first action. This procedure is repeated in a receding-horizon manner until the task is completed.

Decision Diffuser. Decision Diffuser (Ajay et al., 2023) adopts a state-only trajectory formulation and implements CFG by conditioning the diffusion model directly on the normalized episodic return y, rather than an explicit optimality variable. Without a reward predictor for trajectory selection,

DD generates a single trajectory at each inference step. An inverse dynamics model I_ϕ then maps consecutive states to the action to execute, i.e., $a_t = I_\phi(s_t, s_{t+1})$.

AdaptDiffuser. AdaptDiffuser (Liang et al., 2023) extends Diffuser with a self-evolving training pipeline to mitigate the limited diversity of offline RL data. It first trains a Diffuser, generates synthetic expert trajectories, and uses a discriminator to filter high-quality samples for fine-tuning. This process can be iterated to progressively refine the model, with the discriminator defining the direction of self-evolution. At inference time, AdaptDiffuser follows the same procedure as Diffuser.

Hierarchical Diffuser/DiffuserLite. Hierarchical Diffuser (Li et al., 2023) leverages a multi-level architecture that combines jumpy subgoal generation with fine-grained trajectory refinement. Also, DiffuserLite (Dong et al., 2024a) allows for more hierarchy levels and refines only the first interval of the previous layer using a planning refinement process. We again follow their implementations in the `CleanDiffuser` library.

Diffusion Veteran/Prior Guidance. Diffusion Veteran (Lu et al., 2025b) is a simple baseline derived from key design principles such as sampling strategy, backbone architecture, and action generation scheme. Using a Transformer as the architecture, it combines Decision Diffuser with Monte Carlo unconditional sampling with selection rather than CFG. Prior Guidance (Ki et al., 2026) improves DV by learning a prior distribution over the noise space, like Diffusion SRL (Wagenmaker et al., 2025).

C.2.3 Diffusion Policies

Diffusion Behavior Cloning. Diffusion Behavior Cloning (Pearce et al., 2023) models behavior cloning as conditional diffusion, generating actions conditioned on the current observation (and optionally its history). To favor high-likelihood actions within the learned behavior distribution, it introduces Diffusion-X Sampling, which extends the standard denoising process with additional refinement steps. Specifically, after the normal T denoising steps reach $t = 1$, Diffusion-X continues denoising for M additional steps toward higher-likelihood regions.

Diffusion Policy. Similar to Diffusion BC, Diffusion Policy (Chi et al., 2025) models the conditional action distribution with a diffusion policy, but introduces two key designs. First, it adopts closed-loop action chunking, conditioning on the latest T_s observations to predict an H-step action sequence, of which T_a actions are executed before replanning. This improves temporal consistency and action smoothness over single-action prediction. Second, Diffusion Policy supports both CNN-based and Transformer-based architectures. The former conditions action generation on observations via FiLM, while the latter fuses observation and action features through cross-attention. Both architectures predict the diffusion noise conditioned on the observation and diffusion timestep.

Diffusion Steering via Reinforcement Learning. Diffusion SRL (Wagenmaker et al., 2025) performs steering over a frozen pre-trained policy, training a high-level RL policy to effectively modulate the base policy's output. The prior is trained to maximize a value function defined over $\mathbf{x}_T$, which estimates the value of the corresponding denoised sample $\mathbf{x}_0$. This shared mechanism allows both methods to bias sampling toward high-value regions of the data or trajectory space. To reproduce this baseline, we first train a diffusion policy, then take its checkpoints (around 20000 iters) as the initial.

C.3 Metrics

C.3.1 Metric on RL/IL tasks

On the D4RL benchmark, we report the normalized score, which is scaled to the range of 0 to 100 as score $= 100 \times \frac{\text{score}-\text{random_score}}{\text{expert_score}-\text{random_score}}$. For the IL benchmarks, we use task-specific evaluation metrics: target-area coverage for PushT and success rate for Robomimic. In the Relay Kitchen environment, since the vast majority of human demonstrations complete only four subtasks, we denote the success rate of completing the i-th subtask by p_i and report the average success rate as score $= (p_1 + p_2 + p_3 + p_4)/4$.

C.3.2 Metric on Data Quality

In the following, we precisely describe the computation of the test metrics (Liao et al., 2024). Let $(X_t)_{t=1}^T$ denote a d-dimensional sequences sampled from the target distribution. We first extract

input–output pairs $\left(\overline{X}_{t-p+1},,\overline{X}_{t+1+\bar{q}}\right)_{t\in\mathcal{T}}$, where $\mathcal{T}$ denotes the set of valid time indexes. Given a generator G, for each input sample $\overline{X}_{t-p+1}$, we generate one $\bar{q}$-step imputation $\widehat{X}^{(t)}_{t+1+\bar{q}}$. If G is unconditional, the imputation is generated without conditioning on the input sample. The resulting synthetic trajectories $\left(\widehat{X}^{(t)}_{t+1+\bar{q}}\right)_{t\in\mathcal{T}}$ are then used to compute the test metrics.

Temporal dependency: For the real data, the autocovariance of the i-th feature at lag k is defined as

$$\rho^r_i(k) \coloneqq \frac{1}{T-k}\sum_{t=1}^{T-k}\left(X^i_t-\overline{X}^i\right)\left(X^i_{t+k}-\overline{X}^i\right),$$

where $\overline{X}^i$ denotes the mean of the i-th feature over the entire sequence. For the synthetic data, we estimate the auto-covariance of the i-th feature at lag k as

$$\rho^i_G(k) \coloneqq \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\widehat{X}^{(t),i}_{t+1}\widehat{X}^{(t),i}_{t+k+1} - \left(\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\widehat{X}^{(t),i}_{t+1}\right)\left(\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\widehat{X}^{(t),i}_{t+k+1}\right).$$

The lag-1 autocorrelation estimators for the real and synthetic data are respectively given by $\frac{\rho^i_r(1)}{\rho^i_r(0)}$ and $\frac{\rho^i_G(1)}{\rho^i_G(0)}$. We define the ***ACF score*** as the mean absolute difference between the lag-k estimates:

$$\text{ACF score} \coloneqq \frac{1}{d}\sum_{i=1}^{d}\left|\frac{\rho^i_r(k)}{\rho^i_r(0)}-\frac{\rho^i_G(k)}{\rho^i_G(0)}\right|. \tag{29}$$

Spatial dependency: We get the covariance between the i-th and j-th features of the real data as

$$\text{cov}^{i,j}_r = \frac{1}{T}\sum_{t=1}^{T}X^i_tX^j_t - \left(\frac{1}{T}\sum_{t=1}^{T}X^i_t\right)\left(\frac{1}{T}\sum_{t=1}^{T}X^j_t\right).$$

Similarly, for the synthetic data, we estimate the covariance between the i-th and j-th features by averaging over all generated action/state steps:

$$\text{cov}^{i,j}_G = \frac{1}{\bar{q}}\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\sum_{s=1}^{\bar{q}}\widehat{X}^{(t),i}_{t+s}\widehat{X}^{(t),j}_{t+s} - \left(\frac{1}{\bar{q}}\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\sum_{s=1}^{\bar{q}}\widehat{X}^{(t),i}_{t+s}\right)\left(\frac{1}{\bar{q}}\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\sum_{s=1}^{\bar{q}}\widehat{X}^{(t),j}_{t+s}\right).$$

The corresponding correlation coefficients for the real and synthetic data are given by

$$\tau^{i,j}_r \coloneqq \frac{\text{cov}^{i,j}_r}{\sqrt{\text{cov}^{i,i}_r\,\text{cov}^{j,j}_r}}, \qquad \tau^{i,j}_G \coloneqq \frac{\text{cov}^{i,j}_G}{\sqrt{\text{cov}^{i,i}_G\,\text{cov}^{j,j}_G}}.$$

Finally, we measure the discrepancy between the ***correlation*** structures of the real and synthetic data using the ℓ_1 norm of the difference between their correlation matrices:

$$\text{CC score} \coloneqq \left\|\left(\tau^{i,j}_r\right)_{i,j\in\{1,\dots,d\}} - \left(\tau^{i,j}_G\right)_{i,j\in\{1,\dots,d\}}\right\|_1. \tag{30}$$

Distribution fitting: We flatten each sample into a vector of dimension $D = wd$ (w the sequence length, d the number of features), and subsample $N = \min\{N_{\text{real}}, N_{\text{gen}}, 2000\}$ real and synthetic samples. We then fit a multivariate Gaussian to each set, with mean and covariance

$$\mu_r \coloneqq \frac{1}{N}\sum_{n=1}^{N}\overline{X}^{(n)}, \qquad \Sigma_r \coloneqq \frac{1}{N-1}\sum_{n=1}^{N}\left(\overline{X}^{(n)}-\mu_r\right)\left(\overline{X}^{(n)}-\mu_r\right)^\top,$$

and analogously μ_G, Σ_G for the synthetic samples. When D is large ($D > 5000$), we first fit a PCA (Abdi & Williams, 2010) on the real samples and project both sets onto the $k = 256$ leading components, using the same projection for real and synthetic data. We define the ***FID score*** as the Fréchet distance between the two Gaussians,

$$\text{FID score} \coloneqq \|\mu_r-\mu_G\|_2^2 + \text{Tr}\left(\Sigma_r+\Sigma_G-2\left(\Sigma_r^{1/2}\Sigma_G\Sigma_r^{1/2}\right)^{1/2}\right), \tag{31}$$

where covariances are made full-rank via their eigendecomposition with a small $\epsilon = 10^{-6}$ and negative eigenvalues are clipped to zero.

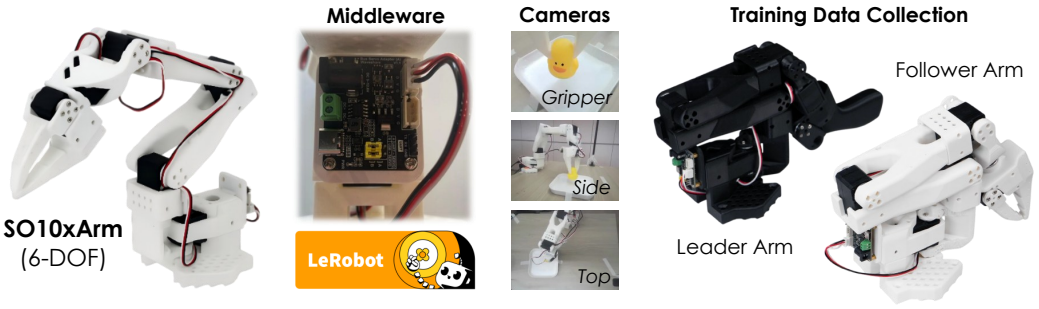


Figure 17: Real-world single-task `lerobot` setup.

Table 3: **Hyperparameters for Diffusion Policy and Diffusion BC in Low-Dim and Image Tasks.**

Hyperparameters	Diffusion Policy			Diffusion BC	
Architecture	`Chi_UNet1d`	`Chi_Transformer`	`DiT1d`	`Pearce_MLP`	`DiT`
Sampling Steps	5 (PushT) 50 (Otherwise)	5 (PushT) 50 (Otherwise)	5 (PushT) 50 (Otherwise)	50	50
Horizon	16	10	10	2	2
Obs Steps	2	2	2	2	2
Action Steps	8	8	8	1	1
Gradient Steps	1e6	1e6	1e6	1e6	1e6
Batch Size	256 (Low dim) 64 (Image)	256 (Low dim) 64 (Image)	256 (Low dim) 64 (Image)	512 (Low dim) 64 (Image)	512 (Low dim) 64 (Image)
Temperature	1.0	1.0	1.0	1.0	1.0
Learning Rate	1e-4	1e-4	1e-4	1e-3	5e-4
Extra Sample Steps	N/A	N/A	N/A	8	8
Control Mode	Pos	Pos	Pos	Vel	Vel

C.4 Physical Testbed

Computing resources. Most experiments are conducted on a Linux server equipped with two Intel(R) Xeon(R) Platinum 8358 CPUs @ 2.60GHz, providing 64 physical cores and 128 hardware threads, and two NVIDIA A800-SXM4-80GB GPUs, each with 80GB HBM2e memory.

Real-world deployment. Our robot manipulation experiments are conducted on the SO-10xARM platform, an open-source robotic arm developed by RobotStudio in collaboration with Hugging Face. The SO-101 (or SO-10X) is equipped with six joints (6-DOF) and three cameras (gripper, top, and side), providing multimodal visual observations for manipulation tasks, which is fully compatible with the open-source LeRobot framework (Cadene et al., 2026). As shown in Fig. 17, `lerobot` is designed to interface multiple open devices with a shared middleware that can be used to (1) read the configuration on a *leader* robot and write it on *follower* robot for teleoperation and (2) directly control the *follower* with a learned policy. For training, we utilize two NVIDIA A100-80GB GPUs.

C.5 Detailed Implementation of KDDM

We use AdamW (Kingma & Ba, 2014) with a learning rate of 1.0×10^{-4}, and apply exponential moving average (EMA) updates with decay rate 0.995. Two different noise schedules are used in **KDDM** and **KDIM** by default: the *Linear Noise Schedule* (Ho et al., 2020) for coupling strength $\{\kappa_t\}_{t=1}^T$ and the *Cosine Noise Schedule* (Nichol & Dhariwal, 2021) for noise strength $\{\alpha_t\}_{t=1}^T$. The former defines $\kappa_t = \exp\left(-\frac{\beta_1-\beta_0}{2}t^2 - \beta_0 t\right)$. The latter schedule defines $\alpha_t = \cos\left(\frac{\pi}{2}\cdot\frac{t+s}{1+s}\right)/\cos\left(\frac{\pi}{2}\cdot\frac{s}{1+s}\right)$, where $s = 0.008$ and $\sigma_t = \sqrt{1-\alpha_t^2}$. For the learned $\mathbf{K}_\psi(\boldsymbol{\theta}_0, t)$, we use an MLP as the underlying network architecture, with Transformer-style sinusoidal positional embeddings (Ho et al., 2020) to encode the diffusion timestep t.

Unless stated otherwise, we utilize default hyperparameters from the official implementations for most offline RL algorithms and datasets in `CleanDiffuser`. For IL algorithms, i.e., Diffusion Policy and Diffusion BC, key hyperparameters used in our experiments are presented in Table 3. The default

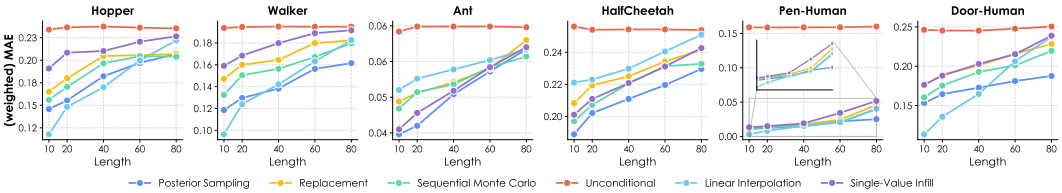


Figure 18: A comparison of the weighted errors for the trajectory stitching methods.

network architectures are `Pearce_MLP` and `Chi_UNet1d`, respectively. Regarding Diffusion SRL, the RL agent only learns a latent actor $\pi_\theta(w \mid f)$ that outputs a noise vector $w \in \mathbb{R}^{H\times d}$ spanning the entire action horizon. Only the executed slice of the decoded action chunk is passed to the environment and scored by the critic. The condition feature f is the flattened output of the frozen EMA encoder, We instantiate two variants: DSRL-NA, where the action critic $Q_A(f, a)$ is trained on the executed normalized action chunk and a separate latent critic $Q_W(f, w)$ supplies the actor gradient, which avoids backpropagating through the diffusion sampling chain; and DSRL-SAC, which applies the SAC update directly in the latent space.

In the real-world SO-ARM100/SO-ARM101 experiments, we trained a diffusion policy proposed by Chi et al. (2025) using the UNet architecture implemented by `lerobot` on self-collected datasets. To be more specific, the policy first encodes observations from multiple cameras using separate ResNet-18 encoders followed by Spatial Softmax, yielding compact visual representations that are concatenated with the robot state to form the global conditioning signal. The resulting condition is fed into a 1D conditional UNet with FiLM-based modulation, which models the temporal structure of the action sequence. We use a horizon of 64 action steps and predict 32 actions at a time, with the latest two observations serving as the conditioning context. During inference, the policy starts from Gaussian noise and iteratively performs the reverse diffusion process to generate an action chunk, from which 32 consecutive actions are executed sequentially.

D Additional Experiments

This section provides additional experiments to deepen our understanding of **KDDM** beyond the main results. We examine its controllable trajectory generation capability, generalization to more image-based manipulation tasks, and the respective contributions of the Kuramoto model and the learned $\mathbf{K}_\psi(\boldsymbol{\theta}_0, t)$. We also investigate how the number of sampling steps affects the quality and efficiency of trajectory generation. Together, these experiments provide a more detailed characterization of the proposed method from the perspectives of controllability, generalization, and sampling efficiency.

D.1 Generative Trajectory Stitching

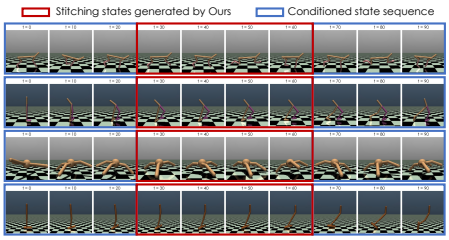


Figure 19: The stitching trajectories.

To assess the controllable generation capability of our proposed approach, we study the trajectory stitching problem (Li et al., 2024a; Luo et al., 2026), which is defined as follows. Consider two trajectories, $\boldsymbol{\tau} = ([\mathbf{s}_0, \mathbf{a}_0], [\mathbf{s}_1, \mathbf{a}_1], \ldots, [\mathbf{s}_H, \mathbf{a}_H])$ and $\boldsymbol{\tau}' = ([\mathbf{s}'_0, \mathbf{a}'_0], [\mathbf{s}'_1, \mathbf{a}'_1], \ldots, [\mathbf{s}'_{H'}, \mathbf{a}'_{H'}])$, where H and H' denote their respective lengths. A straightforward approach to stitching these trajectories is to construct a masked trajectory by concatenating their elements with a sequence of masked entries:

$$([\mathbf{s}_0, \mathbf{a}_0], [\mathbf{s}_1, \mathbf{a}_1], \ldots, [\mathbf{s}_H, \mathbf{a}_H], [\texttt{MASK}], \ldots, [\texttt{MASK}], [\mathbf{s}'_0, \mathbf{a}'_0], [\mathbf{s}'_1, \mathbf{a}'_1], \ldots, [\mathbf{s}'_{H'}, \mathbf{a}'_{H'}]) .$$

The diffusion model is then conditioned on the observed portions of this trajectory and tasked with recovering the masked segment, thereby generating a coherent transition between $\boldsymbol{\tau}$ and $\boldsymbol{\tau}'$.

We simulate this task by taking 300 trajectories from each dataset and masking L intermediate steps in each trajectory. We evaluate three controllable generation schemes: *Replacement*, *Posterior Sampling*, and *Sequential Monte Carlo*. We exclude other training-based strategies, i.e., *CG* and *CFG*,

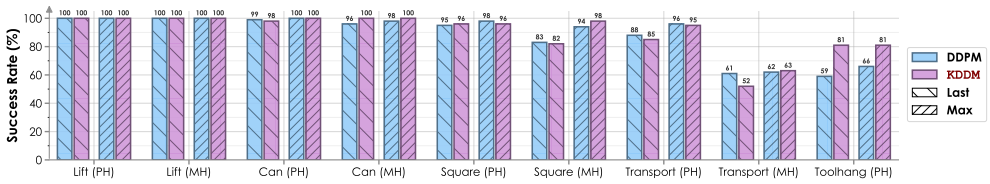


Figure 20: Image-based task success rate of Diffusion Policy with different diffusion backbones.

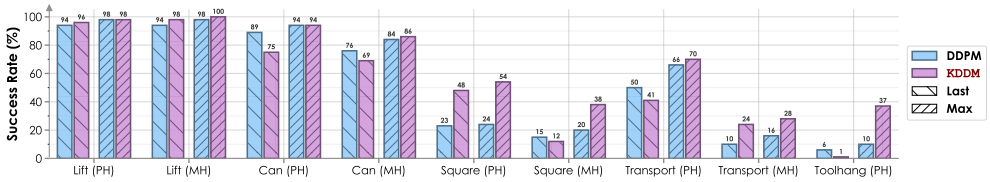


Figure 21: Image-based task success rate of Diffusion BC with different diffusion backbones.

as their effectiveness has been validated in our preceding experiments. As baselines, we consider *unconditional* generation, mean imputation (*Single-Value Infill*), and classical *linear interpolation*. We vary the mask length L over $10, 20, 40, 60, 80$ and evaluate the reconstruction quality using a weighted Mean Absolute Error (MAE), with larger weights assigned to the endpoints to emphasize the continuity of the stitched trajectories. Fig. 18 shows our results for the six methods. We can see that (i) a general inverse correlation is observed between performance and mask length; (ii) linear interpolation yields minimal error when the mask length is short, but the error subsequently increases almost linearly; and (iii) all three of our proposals outperform the unconditional generation baseline, with Posterior Sampling achieving the best performance despite its higher computational cost for gradient calculation. We visualize the robot states at each time step, and the results are presented in Fig. 19. These results demonstrate that our proposed conditioned schemes for **KDDM** are effective.

D.2 More Image-Based Tasks

In addition to PushT-Image in the main text, we further evaluate our method on a suite of image-based tasks from the Robomimic benchmark. Figs. 20 and 21 report the success rates of Diffusion Policy and Diffusion BC across 9 tasks respectively; the results of the DDPM version are obtained from Dong et al. (2024b). Under controlled comparison, we find experimental results similar to those on the corresponding state-based tasks. For Diffusion Policy, performance gaps emerge only on a small subset of tasks requiring high precision (e.g., precise insertion tasks). But for Diffusion BC, **KDDM** yields pronounced performance improvements on half of the tasks. For instance, in terms of the best performance achieved (i.e., MAX), **KDDM** improves the average success rate by 10.6 percentage points across all tasks and by 18.2 percentage points across the last 5 tasks (from left to right). Therefore, our approach remains effective in visual scenarios, as it requires no modifications to the underlying architecture while retaining its advantages in general.

D.3 Effect of Kuramoto Model

To understand the influence of the synchronization process introduced by the Kuramoto Model, we first compare **KDDM** with a variant, **KDDM** w/o Synchronization (or w/o Syn. for short). Specifically, we directly remove the drift term in Eqns. (4) and (6), reducing **KDDM** to a DDPM-like form without dependencies on its forward/reverse process. As seen in Table 4, on the unconditional generation tasks in the Adroit Hand environment, removing the synchronization mechanism leads to degraded data quality across all three metrics. As expected, the ability to capture temporal (ACF) and spatial (CC) dependencies deteriorates more substantially than distribution (FID) fitting.

Table 4: Ablation results on the Adroit Hand datasets.

Dataset	Method	FID Score	ACF Score ($\times 10$)	CC Score
Door-Human	Original	**604.5** $\pm$ 1.48	**17.9** $\pm$ 0.02	**120.0** $\pm$ 5.21
	w/o Syn.	647.7 $\pm$ 1.91	22.5 $\pm$ 0.12	174.9 $\pm$ 3.73
Pen-Human	Original	**353.4** $\pm$ 9.48	**2.58** $\pm$ 0.02	**177.9** $\pm$ 9.83
	w/o Syn.	373.2 $\pm$ 5.44	3.04 $\pm$ 0.01	280.5 $\pm$ 4.45

Since we have conducted data augmentation and imitation learning ablations (in)directly, by replacing the diffusion backbone, we now perform the corresponding experiments on reinforcement learning tasks (though §5.2 already includes an ablation against DV). For broader coverage of different

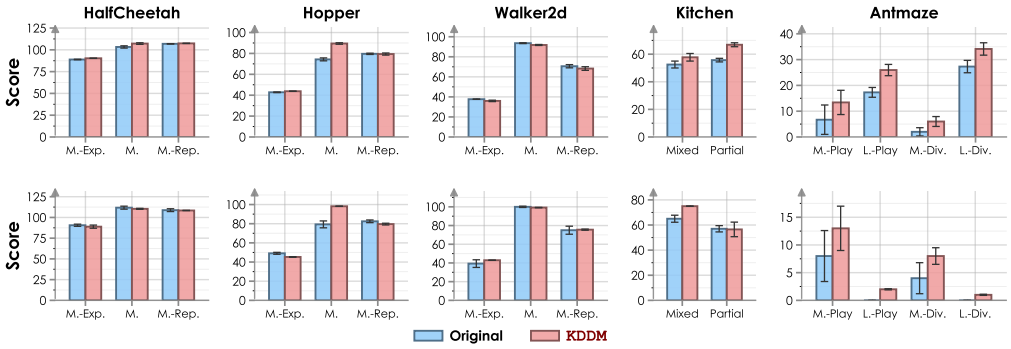


Figure 22: Comparing diffusion-based planning with **KDDM** against the original configuration. The first row corresponds to Diffuser, while the second corresponds to Decision Diffuser.

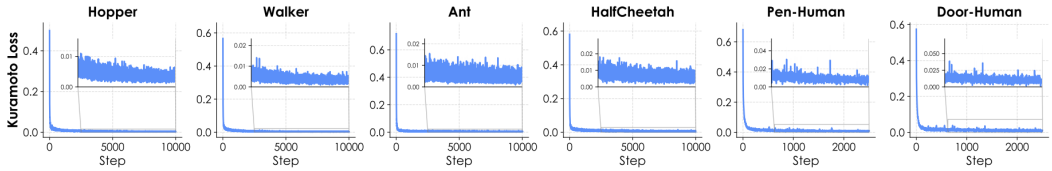


Figure 23: Pretraining ℓ_1-loss value or fitting curves of the learning-based $\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)$.

frameworks, we evaluate on Diffuser and Decision Diffuser. The former uses CG to jointly generate states and actions, whereas the latter uses CFG to generate only states and then infers actions via inverse dynamics. Our results are given in Fig. 22, where we plot the performance of the learned planners averaged over 150 seeds. As this figure illustrates, we see that (de)synchronization through the Kuramoto model provides a consistent improvement over planning across all datasets, suggesting that our physics-inspired scheme is an effective way to incorporate spatio-temporal dependencies into trajectory diffusion models for decision making.

D.4 Effect of Learning-Based $\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)$

Earlier discussion in §4.3 suggested that the Kuramoto coupling term $\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)$ is learnable and potentially further improve the efficiency of **KDDM**. These learned drifts are used as the default setting in our experiments. Examining Fig. 23 that plots the ℓ_1 reconstruction loss curves over pretraining iterations, we can see that, across all datasets, the loss rapidly decreases to a reasonable error range and then stabilizes between 0.01 and 0.04, where it plateaus. In practice, we find that the pre-trained $\mathbf{K}_\psi\left(\boldsymbol{\theta}_0, t\right)$ generalizes well during our desynchronization processes, regardless of the $\hat{\boldsymbol{\theta}}_0$ it receives from the reconstructor as input, with the relative error with respect to the GT Kuramoto model being almost always below 10%. For fixed conditions, we evaluate the relative performance of the learning-based version compared to the non-learning-based version across common benchmarks (i.e., RL and IL); see Fig. 24. For each task, the performance is averaged across 3 seeds. We have the following observations: (i) the learned one-step mapping function accelerates training and inference by around 70% and 40%, delivering a substantial efficiency gain; (ii) the two versions achieve parity (< 8% performance degradation) across the vast majority of decision-making tasks. These results highlight the necessity of pre-training a fast, learning-based Kuramoto model.

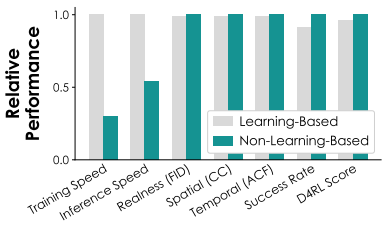


Figure 24: Relative performance comparison across multiple metrics.

D.5 Impact of Sampling Steps

To conclude this section, we examine the impact of the number of sampling steps on trajectory generation quality. Fig. 25 presents the median-normalized results of **KDDM** and **KDIM** with sampling steps $T = 100, 95, 80, 60, 40, 20, 10, 5, 3$. Here, $T = 100$ specifically corresponds to **KDDM**, while the remaining settings use **KDIM** for the reverse process. We make two observations. First, trajectory

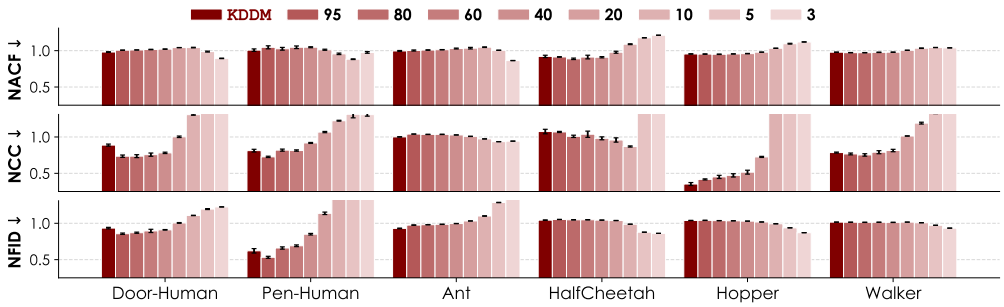


Figure 25: Generative performance of **KDDM** and **KDIM** with varying sampling steps.

quality generally degrades as the number of sampling steps decreases, with excessively few steps potentially causing a sharp drop in performance. Second, some tasks exhibit the opposite trend, where performance decreases as the number of sampling steps increases, a phenomenon known as *sampling degradation*. This behavior has also been reported in prior studies (Chen et al., 2022; Kang et al., 2023; Dong et al., 2024b) and remains an open question. In our experiments, it occurs more frequently in MuJoCo tasks, possibly due to their relatively narrow data distributions. A systematic investigation of this phenomenon and the identification of task-specific sampling steps are left for future work. Notably, around 40 sampling steps are sufficient for most decision-making scenarios, suggesting that substantially more steps, such as 100 (Ajay et al., 2023), are often unnecessary.

E VISUALIZATIONS

This section provides visual evidence of the behaviors and capabilities of **KDDM**. We begin with the (de)synchronization processes to illustrate how the proposed mechanism shapes the diffusion dynamics, and then visualize the learned distributions. Finally, we present visualizations of robotic manipulation in both simulation and real-world environments.

E.1 (DE)SYNCHRONIZATION PROCESSES

We first visualize the (de)synchronization phenomena of our Kuramoto model. Using six representative RL and IL datasets, this is shown in Fig. 26, where we illustrate the Kuramoto dynamics on a unit circle, and trajectories are mapped onto periodic domains as angular phase variables. As can be seen, a large cluster of synchronous oscillators is apparent over time. We also observe that initial synchronization levels vary across tasks, ranging from 0.3 to 0.7, yet the order parameter r stably increases to 1. In this test, convergence is typically achieved within 60 steps, while increasing the coupling strength can substantially accelerate convergence, often to fewer than 10 steps.

We next visualize the diffusion dynamics. As shown in Fig. 27, we compare the marginal distributions of DDPM and **KDDM** by applying the forward process to the same sample. Although both methods progressively approach the real distribution, our method gets "into the groove" noticeably earlier. This observation is consistent with the empirical SNR analysis in Fig. 6 of Song et al. (2026), which shows that coupling-driven dynamics enable a more efficient and structured progression of noise.

E.2 LEARNED DISTRIBUTIONS

To visually demonstrate the distribution fitting ability, we conduct a visualization analysis of trajectories generated by **KDDM** and KODM using the t-SNE and PCA methods. The resulting two-dimensional embeddings are shown in Fig. 28 and Fig. 29, respectively. From the figures, we observe the following: (1) KODM learns a relatively narrow distribution that is readily distinguishable from the real samples, making it less capable of capturing multimodal scenarios. (2) Samples generated by **KDDM** exhibit substantial overlap with the real data while keeping well dispersed, suggesting better generalization and higher data quality than KODM. These observations can be explained by Proposition 2.

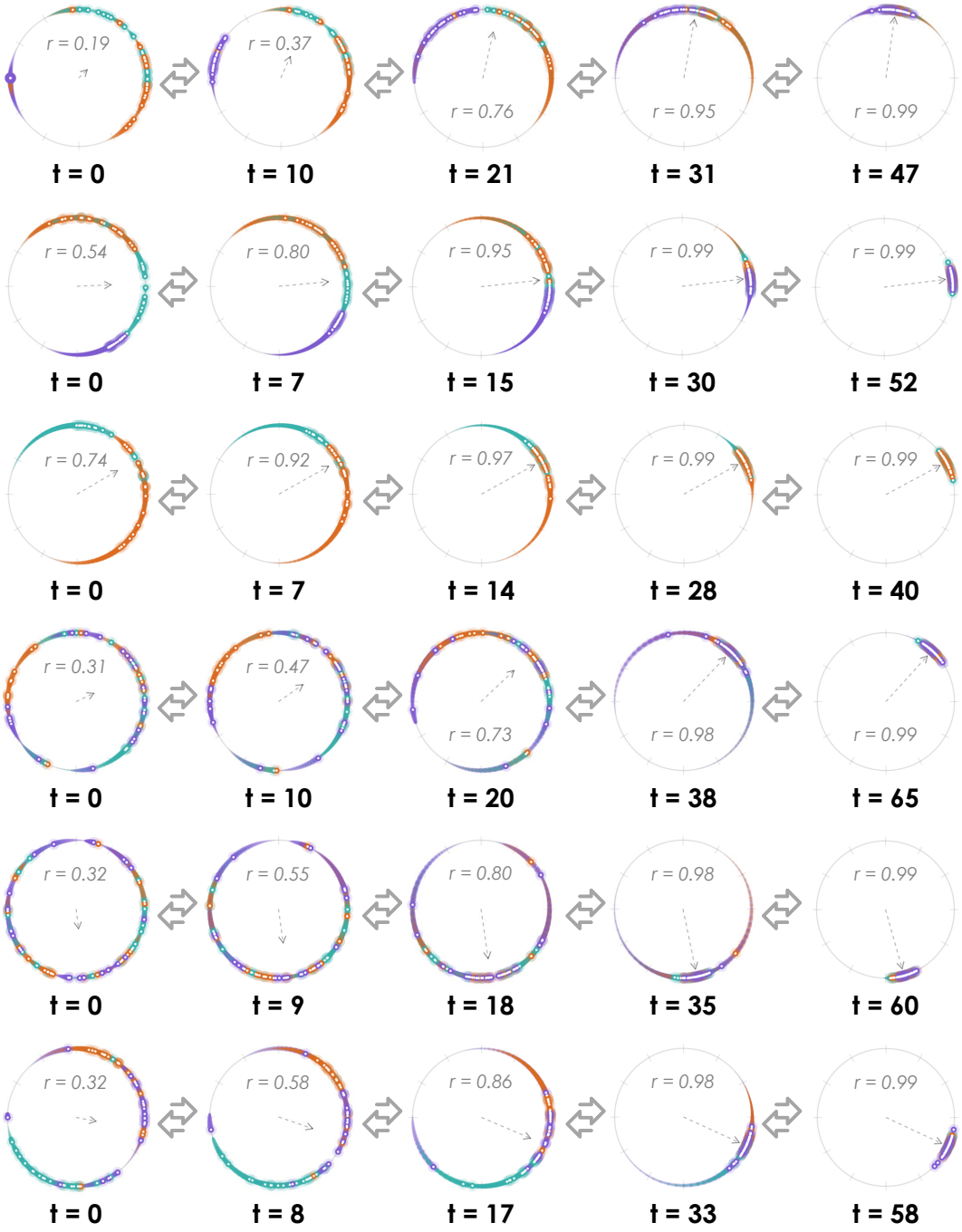


Figure 26: Visualizations of the synchronization processes using the Kuramoto model in robotic trajectories. From the first to the sixth row, the subfigures correspond to *tool-hang-ph*, *can-mh*, *PushT*, *Hopper-Medium*, *Antmaze-Medium-Play*, and *relay-kitchen*, respectively. These trajectories are chunked and randomly sampled from our dataset. Oscillators or points corresponding to the same feature (e.g., action or/and state) at different time steps are assigned the same color. For clarity, we visualize only the three dimensions exhibiting the most pronounced changes, except for *PushT*, which contains only two such dimensions.

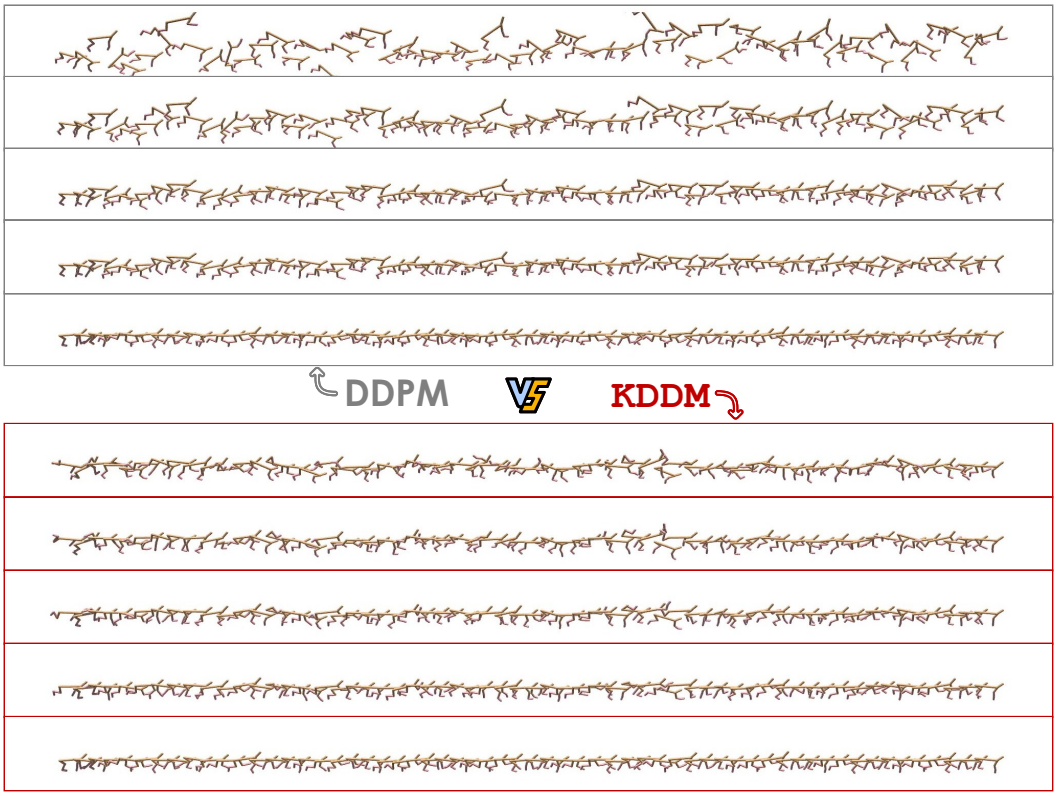


Figure 27: Comparison of diffusion processes on the HalfCheetah dataset for DDPM (top) and **KDDM** (bottom). Following Janner et al. (2022), we simulate the forward process for one data sample and render these noisy trajectories in the MuJoCo environment. The visualizations are arranged from top to bottom in descending order of diffusion timestep t. Also note that the two diffusion schemes are shown at the same diffusion timesteps in each row.

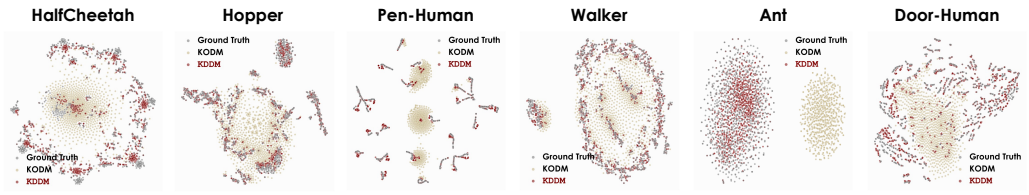


Figure 28: Visualization of trajectory distributions using the *t-SNE* method. In all plots, gray dots represent real data instances, while yellow and red dots represent trajectories generated by KODM and **KDDM**, respectively. Better coverage or being closer to the GT indicates better distribution modeling.

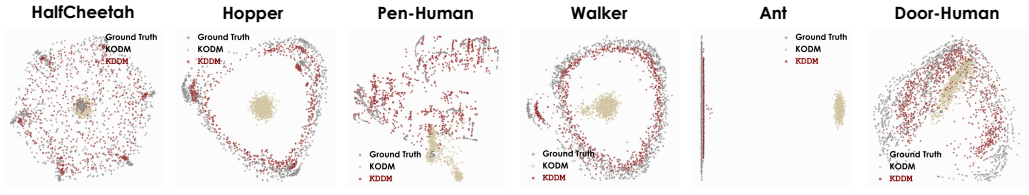


Figure 29: Visualization of trajectory distributions using the *PCA* method. In all plots, gray dots represent real data instances, while yellow and red dots represent trajectories generated by KODM and **KDDM**, respectively. Better coverage or closer to the GT indicates better distribution modeling.



Figure 30: Planning visualizations for Push-T on randomly sampled initial and goal configurations.

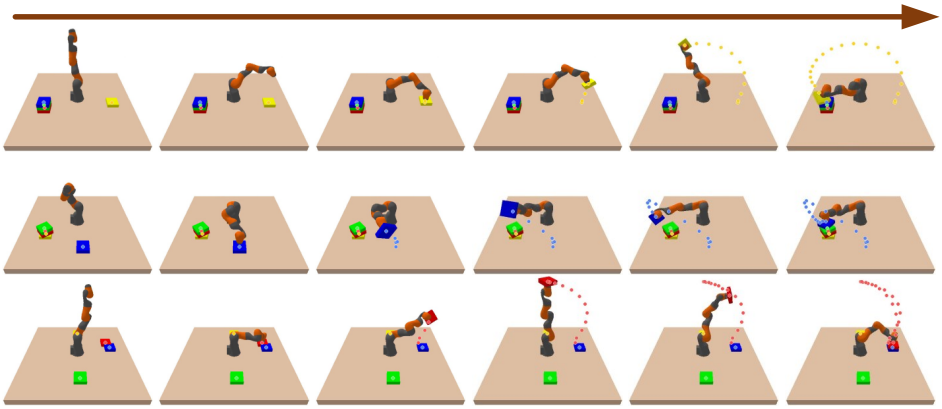


Figure 31: Rendering of the KUKA robotic manipulation environment. We show six frames from a single trajectory solving both the conditional and unconditional block stacking task.

E.3 Examples of Simulational Manipulation

We visualize representative manipulation trajectories generated by **KDDM** across several simulation environments, covering planning, composite manipulation, and multi-stage robotic tasks. Specifically, Fig. 30 presents planning visualizations for Push-T, Fig. 31 shows a KUKA block-stacking trajectory, Fig. 32 illustrates composite manipulation tasks on Relay Kitchen, and Fig. 33 provides example demonstration trajectories from three Robomimic IL tasks. Together, these visualizations provide a qualitative view of the decisions generated by **KDDM** across different decision-making settings.

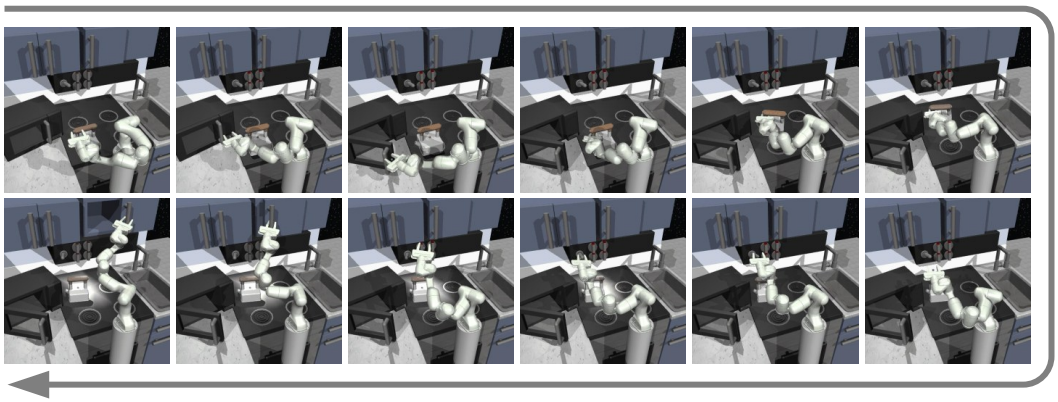


Figure 32: Decision-making visualizations for composite manipulation tasks on Relay Kitchen.

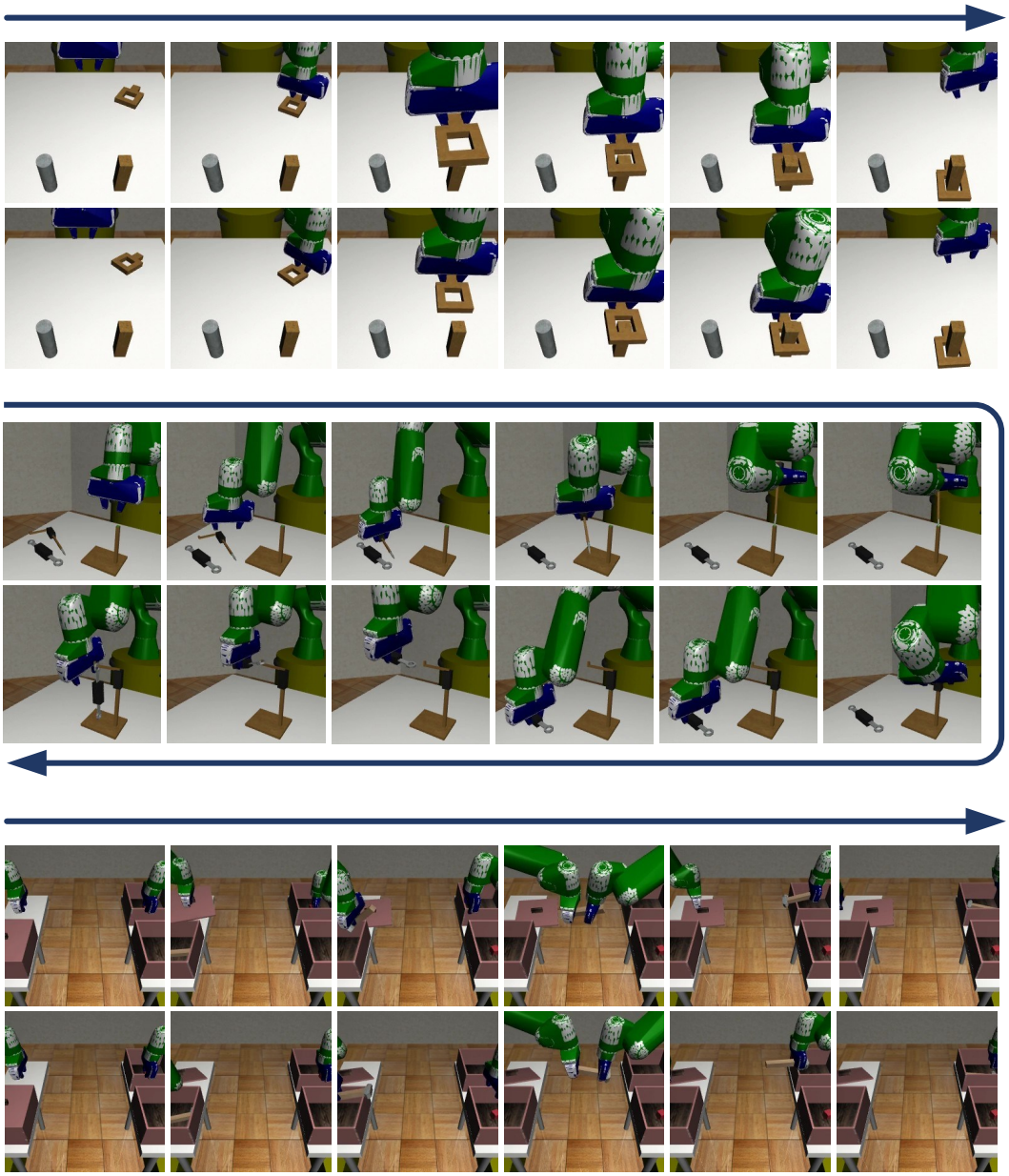


Figure 33: Manipulation task demonstrations on Robomimic benchmark, covering pick-and-place, multi-arm coordination, and high-precision insertion and assembly. Here we showcase example demonstration trajectories for the Square (PH) (top), Tool Hang (PH) (middle), and Transport (PH) (bottom) tasks, illustrating the different stages of each task using an offline diffusion policy with **KDDM** as the diffusion backbone.

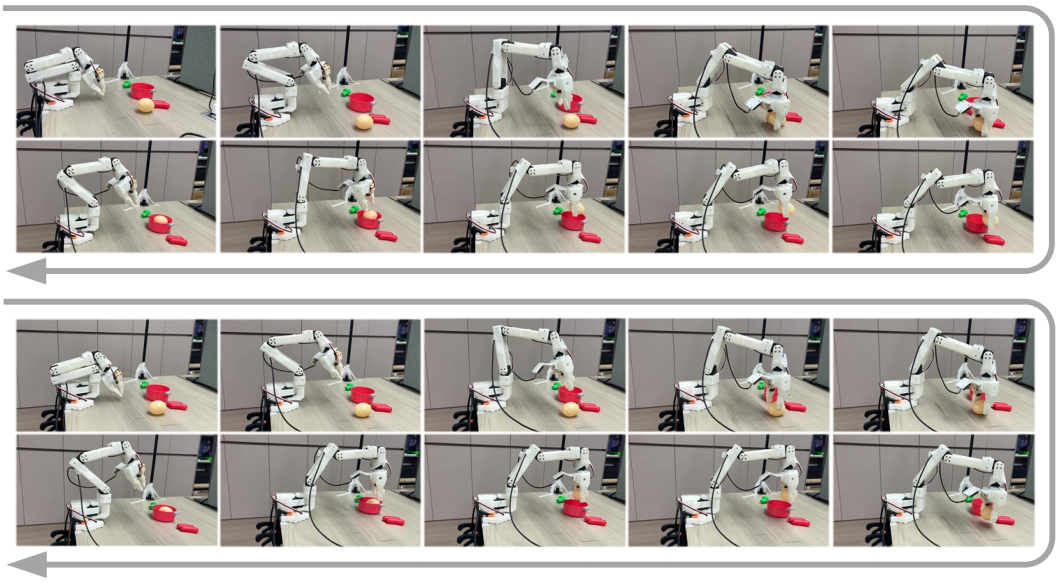


Figure 34: Executed rollout demonstrations on a real-world task, "`pick up the egg and put it in the pot`", for **KDDM**-backboned diffusion policy in the LeRobot environment using a 6-DOF robotic arm.

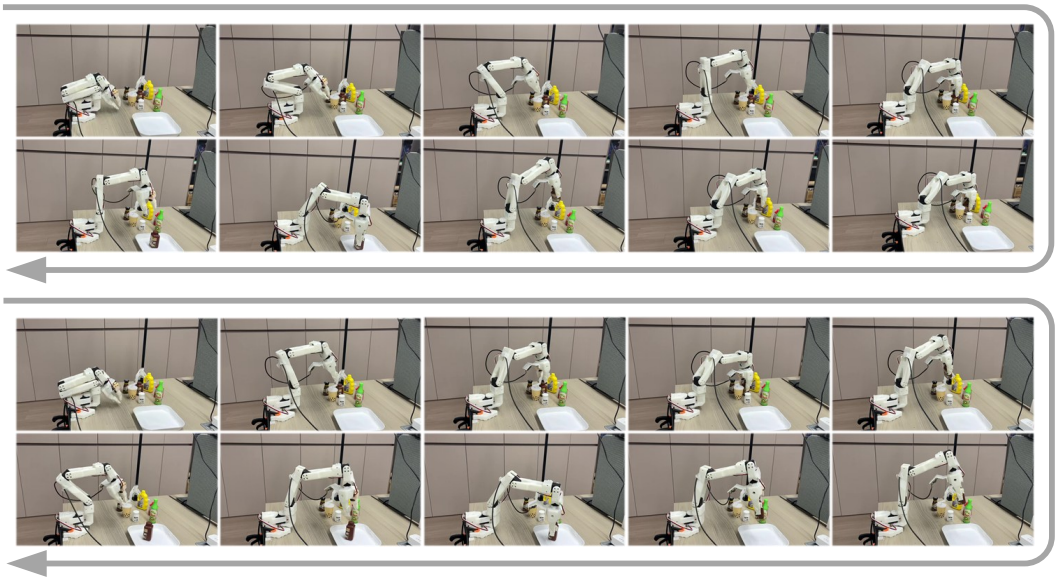


Figure 35: Executed rollout demonstrations on a real-world task, "`pick up the brown bottle and place it on the plate`", for **KDDM**-backboned diffusion policy in the LeRobot environment using a 6-DOF robotic arm.

E.4 Examples of Real-World Manipulation

Finally, we visualize the real-world validation of **KDDM** across four test scenes, `put-the-egg`, `pick-the-bottle`, `place-the-duck`, and `put-the-knife`, using an SO-ARM100/SO-ARM101 robotic arm. The corresponding demonstrations are presented in Fig. 34, Fig. 35, Fig. 36, and Fig. 37, respectively. The policy observes the two most recent observations, predicts an action chunk of 64 steps, and executes the first 32 actions in a receding-horizon manner before generating the next chunk. These decision-making tasks all succeed in $\leq$15 seconds.

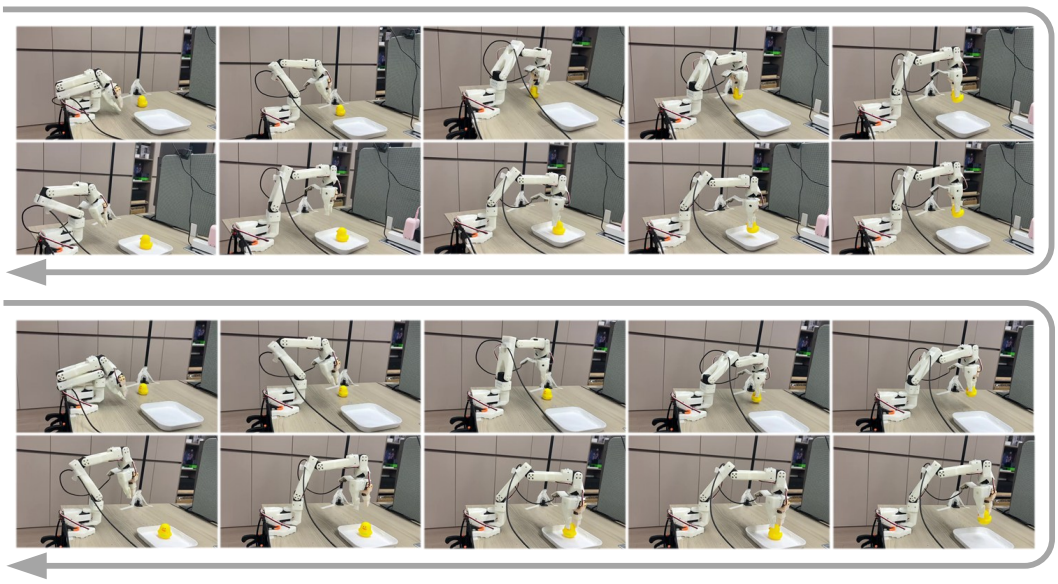


Figure 36: Executed rollout demonstrations on a real-world task, "`pick up the toy duck and place it in the center of the plate`", for **KDDM**-backboned diffusion policy in the LeRobot environment using a 6-DOF robotic arm.

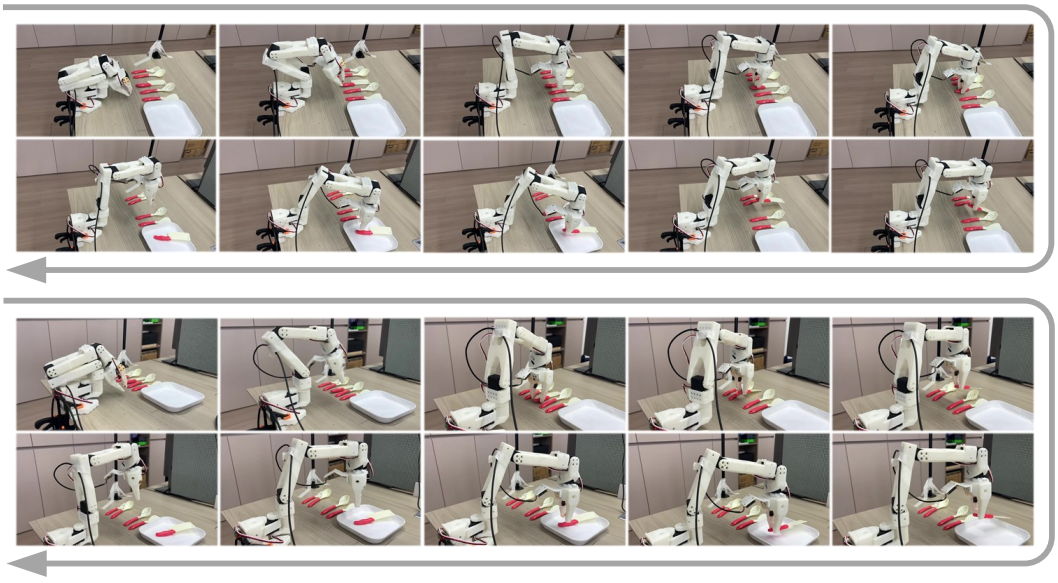


Figure 37: Executed rollout demonstrations on a real-world task, "`place the knife in the center of the plate`", for **KDDM**-backboned diffusion policy in the LeRobot environment using a 6-DOF robotic arm.

F LIMITATIONS

While our results have demonstrated the potential of Kuramoto Decision Diffusion Model (**KDDM**) for decision-making, our study remains subject to several limitations, which we discuss below.

L1 **Relatively slow per-step inference.** Our first trick, *self-synchronization*, needs to compute the coupling term $\mathbf{K}_t$ from 0 to t from scratch in every forward or reverse process. Although learning a mapping largely alleviates this issue, it still introduces an additional forward pass at every diffusion timestep. In our experiments, one iteration of **KDDM** takes approximately twice as long to compute as one iteration of DDPM.

L2 **Large-scale oscillator coordination.** **KDDM** works well on most decision-making tasks, where the number of actions and states are generally below 30, but higher-dimensional settings involve a larger number of oscillators, which may slow down convergence and, in extreme cases, prevent it altogether. For example, the Humanoid task in D4RL has over 300 dimensions, resulting in more than 30,000 coupled oscillators for a trajectory chunk of length 100.

L3 **Interpretability of the used NN.** We directly adapt the underlying model from prior works as the reconstructor. However, according to Eqn. (5), the reconstructor is essentially also attempting to reduce the drift term. A more specialized and interpretable NN for **KDDM** could potentially lead to faster training and/or better quality solutions.

G FUTURE WORKS

Our investigation of the new physics-inspired diffusion paradigm, i.e., **KDDM**, also opens up several intriguing directions for future research, which we outline below.

F1 **More aggressive acceleration.** While we have developed **KDIM** as a DDIM-style ODE solver for *soft* acceleration, exploring PF-ODE (Lai et al., 2025) could provide another avenue for further accelerating the sampling process. For *hard* acceleration, the hierarchical generation process (Rissanen et al., 2022) could potentially trade additional computation and memory for greater parallelism.

F2 **Scalable and local coupling.** A natural variant of the Kuramoto model introduces local coupling, allowing each oscillator to interact only with its neighboring oscillators (Song et al., 2026; Dai & Song, 2026). Such localized interactions improve scalability while retaining diffusion-like dynamics, with oscillatory phases progressively smoothing out across space and time.

F3 **Better-suited NN architectures.** To incorporate the inherent periodicity of phase variables into the reconstructor, one could embed the input phases using sinusoidal features, as in periodicity-aware networks (Song et al., 2026). In addition, a decomposable reconstructor that separately models the coupling and denoising components could be trained jointly and executed sequentially.