Test-Time Sequential Steering of Diffusion Models via Preconditioned Crank-Nicolson

Joel A. Keller[1*] **Taos Transue**[1*] **Qin Li**[2] **Shih-Hsin Wang**[3] **Bao Wang**[1]
[1]Department of Mathematics and Scientific Computing and Imaging Institute, University of Utah
[2]Department of Mathematics, University of Wisconsin–Madison
[3]Department of Computer Science and Information Engineering,
and AI Center of Research Excellence, National Taiwan University

Abstract

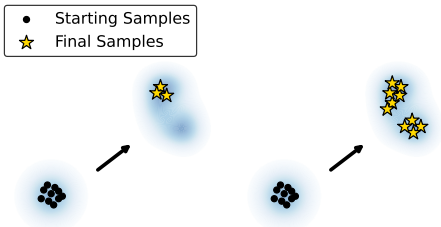
Sampling from reward-tilted distributions enables diffusion models to satisfy task-specific constraints without retraining. However, existing methods either rely on gradients or struggle to explore multimodal high-reward regions. We propose a guidance mechanism based on preconditioned Crank-Nicolson (pCN) Markov chain Monte Carlo, which tilts the reverse transition toward high-reward regions. To further improve exploration in multimodal reward landscapes, we extend this procedure with parallel tempering across a temperature ladder, allowing controlled mixing between exploration and refinement regimes. Our method operates with or without reward gradients, applying to both differentiable and black-box rewards. Empirically, we demonstrate its effectiveness on synthetic tasks, image generation, dynamical systems, and noiseless inverse problems. Code is available at `https://github.com/Utah-Math-Data-Science/RpCN`.

1 Introduction

Advances in generative modeling, especially the advent of diffusion models (cf. [41, 19, 42]), enable sampling from complex, high-dimensional data distributions $p(\boldsymbol{x})$. While these models learn expressive data priors, many downstream tasks require sampling from task-dependent distributions that differ from the learned data distribution. A common formulation is sampling from a *reward-tilted* distribution [30]

$$\tilde{p}(\boldsymbol{x}) \propto p(\boldsymbol{x}) \exp(\beta R(\boldsymbol{x})) \tag{1}$$

where $R(\boldsymbol{x})$ encodes a task-specific reward and $\beta > 0$ controls the strength of tilting. This perspective also encompasses Bayesian inverse problems [43, 25, 45] where, for degradation operator $\boldsymbol{A}$ and noise $\boldsymbol{\xi}$, we aim to recover $\boldsymbol{x}$ from a noisy observation $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{\xi}$ by sampling from the posterior distribution $p(\boldsymbol{x}|\boldsymbol{y}) \propto p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x}) = p(\boldsymbol{x}) \exp(\beta R(\boldsymbol{x}))$ where $R(\boldsymbol{x}) = \log p(\boldsymbol{y}|\boldsymbol{x})/\beta$ is the log-likelihood. Assuming $\boldsymbol{\xi}$ is Gaussian noise, $R(\boldsymbol{x}) \propto -\|\boldsymbol{y} - \boldsymbol{A}(\boldsymbol{x})\|^2$.



(a) FK Steering (b) RpCN (Ours)

Figure 1: Contrasting FK Steering and RpCN for transporting a Gaussian to a multimodal target. RpCN recovers multiple modes, while FK Steering collapses to one.

Existing sampling methods for reward-tilted distributions include training-time and test-time approaches. Training-time methods encode reward preferences into the model parameters, e.g., through fine-tuning or reinforcement learning such as RLHF [37]. Related methods such as Classifier-Free Guidance [20] also rely on jointly trained conditional and unconditional models. While effective, changing the reward requires retraining or further optimization.

*Equal contribution. Corresponding to: wangbaonj@gmail.com

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



Figure 2: Comparison between FK Steering and RpCN (ours) on preference-aligned image generation using Stable Diffusion v1.5 with ImageReward [50]. Prompts are shown in the third row. FK Steering collapses to a limited set of modes and exhibits reduced diversity, whereas RpCN generates more diverse samples while maintaining strong prompt alignment.

In contrast, test-time guidance freezes the pretrained model and modifies the sampling dynamics at inference time, allowing a single model to support diverse downstream rewards without retraining. Existing approaches, such as Classifier Guidance [8], Diffusion Posterior Sampling (DPS) [6], and other gradient-based methods (cf. [42, 1]), provide this flexibility but rely on differentiable rewards and require careful guidance scaling across noise levels to ensure stability [6, 18]. These limitations motivate the development of more robust and flexible test-time guidance methods that can accommodate general rewards while maintaining stability and sample quality.

Gradient-free methods (cf. [40, 17, 49, 47]) partially address this limitation but introduce new challenges. Among them, Feynman–Kac (FK) Steering [40] is specifically designed for diffusion models, sampling tilted transition distributions via Sequential Monte Carlo (SMC) (cf. [35, 5]). In this framework, a population of particles is iteratively denoised and resampled according to reward-dependent weights (see Section 2.2 for details). While FK Steering effectively concentrates probability mass in high-reward regions, repeated resampling can lead to the population collapsing onto a few local modes and thereby reducing distributional coverage, as illustrated in Fig. 1(a). We further highlight this effect on test-time image reward steering tasks in Fig. 2 (see Section 5.2 for experimental details), where FK Steering exhibits pronounced mode collapse.

In practice, the reward-tilted distribution $\tilde{p}(\boldsymbol{x})$ is often multimodal. In such settings, methods that collapse onto local modes—even if they yield high reward—may fail to capture other high-reward regions or find diverse solutions. This raises the following question:

Can we develop a test-time algorithm that accommodates general reward functions while faithfully capturing the full diversity of high-reward regions?

1.1 Our Contribution

In response to the above question, we propose *Residual Preconditioned Crank–Nicolson Steering* (RpCN), a new test-time guidance method for diffusion models that replaces the population-based resampling in FK Steering with local Markov Chain Monte Carlo (MCMC) updates. Specifically, RpCN applies the preconditioned Crank–Nicolson (pCN) operator [3, 7] to the Gaussian reverse transition (see Section 3). To further enhance coverage in multimodal reward landscapes, we incorporate parallel tempering (PT) [11, 44] across a temperature ladder.

We establish a KL divergence bound between the generated distribution and the reward-tilted target. The bound decomposes across diffusion steps and depends on the per-step score matching [24, 23] error and the potential error (Section 4). We evaluate RpCN on image generation, dynamical systems, and inverse problems (Section 5). On images, RpCN achieves, on average, $3\times$ higher diversity than the baseline FK Steering at comparable reward levels. As illustrated in Figs. 1 and 2, FK Steering collapses to a few modes, whereas RpCN maintains strong reward alignment while producing more diverse samples.

2 Background

In this section we provide a brief review on some crucial background materials, including diffusion models, FK Steering for diffusion SMC guidance, and the pCN MCMC sampler.

2.1 Diffusion Models

Diffusion models consist of a forward process that transforms data into noise and a reverse process that inverts this corruption to generate samples. The forward process starts from clean data $\boldsymbol{x}_0 \sim p_{\mathrm{data}}(\boldsymbol{x}_0)$ and has Gaussian marginals

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}\big(\boldsymbol{x}_t;\, \alpha_t \boldsymbol{x}_0,\, \sigma_t^2 \boldsymbol{I}\big), \qquad t = 0, \dots, T,$$

specified by a signal scale $\alpha_t > 0$ and a noise scale $\sigma_t \geq 0$, typically with $\alpha_0 = 1$ and $\sigma_0 \approx 0$. These marginals are then realized by the Markov chain

$$q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0) = \prod_{t=0}^{T-1} q(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t),$$
$$q(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) = \mathcal{N}\big(\boldsymbol{x}_{t+1};\, \alpha_{t+1|t} \boldsymbol{x}_t,\, \sigma_{t+1|t}^2 \boldsymbol{I}\big),$$

where $\alpha_{t+1|t} = \frac{\alpha_{t+1}}{\alpha_t}$, $\sigma_{t+1|t}^2 = \sigma_{t+1}^2 - \alpha_{t+1|t}^2 \sigma_t^2$, i.e., each transition rescales the state and adds Gaussian noise. We write $p_t(\boldsymbol{x}_t)$ for the marginal of $\boldsymbol{x}_t$, so that $p_0(\boldsymbol{x}_0) \approx p_{\mathrm{data}}(\boldsymbol{x}_0)$ when $\sigma_0 \approx 0$. As t increases, $\boldsymbol{x}_t$ loses information about $\boldsymbol{x}_0$, and when α_T is small relative to σ_T, p_T is close to $\mathcal{N}(\mathbf{0}, \sigma_T^2 \boldsymbol{I})$.

The reverse process defines a joint distribution

$$p(\boldsymbol{x}_{0:T}) = p_T(\boldsymbol{x}_T) \prod_{t=0}^{T-1} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}),$$

which induces a generative model over clean data by marginalizing out intermediate states: $p_0(\boldsymbol{x}_0) = \int p(\boldsymbol{x}_{0:T})\, \mathrm{d}\boldsymbol{x}_{1:T}$. In practice, clean data is obtained by first sampling $\boldsymbol{x}_T \sim p_T(\boldsymbol{x}_T) = \mathcal{N}(\mathbf{0}, \sigma_T^2 \boldsymbol{I})$ and then iteratively sampling $\boldsymbol{x}_t \sim p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})$ down to $t = 0$, where each reverse transition is approximated by a Gaussian:

$$p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) \approx \mathcal{N}\big(\boldsymbol{x}_t;\, \boldsymbol{\mu}(\boldsymbol{x}_{t+1}, t+1),\, \tau_{t+1}^2 \boldsymbol{I}\big), \tag{2}$$
$$\boldsymbol{\mu}(\boldsymbol{x}_{t+1}, t+1) = a_{t+1} \boldsymbol{x}_{t+1} + g_{t+1} \nabla \log p_{t+1}(\boldsymbol{x}_{t+1}). \tag{3}$$

The scalars a_{t+1}, g_{t+1} and $\tau_{t+1} > 0$ are determined by the schedule (α_t, σ_t) and sampler. The score is approximated by a neural network $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$ to give the learned reverse transition $p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})$ and its mean $\boldsymbol{\mu}_\theta(\boldsymbol{x}_{t+1}, t+1)$.

2.2 FK Steering

FK Steering [40] samples $\tilde{p}_0(\boldsymbol{x}_0) \propto p_0(\boldsymbol{x}_0) \exp\big(\beta R(\boldsymbol{x}_0)\big)$ via local, time-dependent reweighting along the reverse diffusion trajectory. It constructs an FK path measure that assigns a higher weight to trajectories likely to achieve a high terminal reward. Let $\{G_t(\boldsymbol{x}_{t:T})\}_{t=0}^T$ be a sequence of nonnegative potentials on partial trajectories such that

$$\prod_{t=T}^{0} G_t(\boldsymbol{x}_{t:T}) = \exp\big(\beta R(\boldsymbol{x}_0)\big). \tag{4}$$

The path distribution at time t is weighted by these as follows:

$$p_t^{\mathrm{FK}}(\boldsymbol{x}_{t:T}) \propto p(\boldsymbol{x}_{t:T}) \prod_{s=T}^{t} G_s(\boldsymbol{x}_{s:T}) = p(\boldsymbol{x}_t | \boldsymbol{x}_{t+1})\, G_t(\boldsymbol{x}_{t:T})\, p_{t+1}^{\mathrm{FK}}(\boldsymbol{x}_{t+1:T}).$$

By equation 4, integrating $p_0^{\mathrm{FK}}(\boldsymbol{x}_{0:T})$ over $\boldsymbol{x}_{1:T}$ yields

$$p_0^{\mathrm{FK}}(\boldsymbol{x}_0) = \int p_0^{\mathrm{FK}}(\boldsymbol{x}_{0:T})\, \mathrm{d}\boldsymbol{x}_{1:T} \propto \int p(\boldsymbol{x}_{0:T}) \exp\big(\beta R(\boldsymbol{x}_0)\big)\, \mathrm{d}\boldsymbol{x}_{1:T}$$
$$= \exp\big(\beta R(\boldsymbol{x}_0)\big) \int p(\boldsymbol{x}_{0:T})\, \mathrm{d}\boldsymbol{x}_{1:T} = p(\boldsymbol{x}_0) \exp\big(\beta R(\boldsymbol{x}_0)\big),$$

recovering the tilted marginal $\tilde{p}_0(\boldsymbol{x}_0)$. FK Steering samples $p_0^{\mathrm{FK}}(\boldsymbol{x}_{0:T})$ via SMC. An initial ensemble of particles are sampled from $p_T^{\mathrm{FK}}(\boldsymbol{x}_T)$. For each time step t, the particles are resampled according to their weight $G_t(\boldsymbol{x}_{t:T})$ relative to the ensemble, replicating high-weight trajectories and pruning low-weight ones. The ensemble at time step t approximates the FK path measure $p_t^{\mathrm{FK}}(\boldsymbol{x}_{t:T})$, and at $t = 0$, yields samples from the reward-tilted distribution $\tilde{p}(\boldsymbol{x}_0)$; however, particles tend to collapse into high-reward regions, hindering the exploration of multimodal reward landscapes.

Remark 2.1. *Equation 4 can be realized via different factorizations of the terminal reward into potentials $G_t(\boldsymbol{x}_{t:T})$. One factorization is the difference potential [40] where*

$$G_t(\boldsymbol{x}_{t:T}) = \exp\big(\beta\big(R_t(\boldsymbol{x}_t) - R_{t+1}(\boldsymbol{x}_{t+1})\big)\big) \implies \prod_{t=T}^{0} G_t(\boldsymbol{x}_{t:T}) = \exp\big(\beta R_0(\boldsymbol{x}_0)\big),$$

defining $\{R_t(\boldsymbol{x}_t)\}_{t=0}^T$ as intermediate rewards such that $R_0(\boldsymbol{x}_0) = R(\boldsymbol{x}_0)$ and $G_T(\boldsymbol{x}_T) = 1$. This choice favors trajectories with increasing intermediate reward.

2.3 Preconditioned Crank-Nicolson

Standard random-walk Metropolis methods [16] propose updates of the form $\hat{\boldsymbol{x}} = \boldsymbol{x} + \lambda\boldsymbol{\xi},\ \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I})$, to generate samples of $\pi(\boldsymbol{x})$ with Metropolis–Hastings acceptance probability

$$\alpha(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \min\left\{1, \frac{\pi(\hat{\boldsymbol{x}})}{\pi(\boldsymbol{x})}\right\}.$$

In high dimensions, such proposals suffer from poor scaling: to maintain a non-negligible acceptance rate, the step size must scale as $\lambda = \mathcal{O}(d^{-1/2})$, leading to slow mixing. In the case when

$$\pi(\boldsymbol{x}) \propto \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I})h(\boldsymbol{x})$$

for some nonnegative potential $h(\boldsymbol{x})$, the pCN algorithm [3, 7] addresses this by constructing proposals that preserve the Gaussian reference measure. For $\lambda \in (0, 1)$, the pCN proposal is given by

$$\hat{\boldsymbol{x}} = \sqrt{1 - \lambda^2}\,\boldsymbol{x} + \lambda\boldsymbol{\xi}, \qquad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I}),$$

which is reversible with respect to the Gaussian prior. Thus, the acceptance probability simplifies to

$$\alpha(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \min\left\{1, \frac{h(\hat{\boldsymbol{x}})}{h(\boldsymbol{x})}\right\},$$

By aligning proposals with the reference measure's geometry, pCN avoids vanishing step sizes and maintains stable acceptance rates in high dimensions.

3 RpCN: Our Proposed Algorithm

We present RpCN algorithm in this section. In Section 3.1, we define the reverse transition kernels that bridge to the tilted distribution $\tilde{p}(\boldsymbol{x}) \propto p(\boldsymbol{x}) \exp(\beta R(\boldsymbol{x}))$, and then develop a pCN-based sampler for the kernels in Section 3.2, which we combine with PT for efficient exploration in Section 3.3.

3.1 Intermediate Potential

We aim to sample from the tilted distribution $\tilde{p}_0(\boldsymbol{x}_0) \propto p_0(\boldsymbol{x}_0) \exp\big(\beta R(\boldsymbol{x}_0)\big)$, where p_0 is the reverse diffusion marginal at $t = 0$. Equivalently, we consider sampling the tilted *trajectory* distribution

$$\tilde{p}(\boldsymbol{x}_{0:T}) = \frac{1}{Z} p(\boldsymbol{x}_{0:T}) \exp(\beta R(\boldsymbol{x}_0)), \quad Z = \mathbb{E}_{p_0(\boldsymbol{x}_0)}[\exp(\beta R(\boldsymbol{x}_0))],$$

where $p(\boldsymbol{x}_{0:T})$ denotes the joint distribution of the reverse process with transition kernels $p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$.

For $t \in \{T, \ldots, 0\}$, define the intermediate potential

$$h_t(\boldsymbol{x}_t) = \mathbb{E}_{p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\exp(\beta R(\boldsymbol{x}_0))].$$

Marginalizing the tilted trajectory distribution over $\boldsymbol{x}_{0:t-1}$ gives

$$\begin{aligned}\tilde{p}(\boldsymbol{x}_{t:T}) &= \int \tilde{p}(\boldsymbol{x}_{0:T})\,\mathrm{d}\boldsymbol{x}_{0:t-1} = \frac{1}{Z}\int p(\boldsymbol{x}_{0:T})\exp(\beta R(\boldsymbol{x}_0))\,\mathrm{d}\boldsymbol{x}_{0:t-1} \\ &= \frac{1}{Z}p(\boldsymbol{x}_{t:T})\int p(\boldsymbol{x}_{0:t-1}|\boldsymbol{x}_{t:T})\exp(\beta R(\boldsymbol{x}_0))\,\mathrm{d}\boldsymbol{x}_{0:t-1} \\ &= \frac{1}{Z}p(\boldsymbol{x}_{t:T})\int p(\boldsymbol{x}_0|\boldsymbol{x}_t)\exp(\beta R(\boldsymbol{x}_0))\,\mathrm{d}\boldsymbol{x}_{0:t-1} = \frac{1}{Z}p(\boldsymbol{x}_{t:T})h_t(\boldsymbol{x}_t)\end{aligned}$$

For $t < T$, the Markov factorization of the reverse-diffusion trajectory yields

$$\tilde{p}(\boldsymbol{x}_{t:T}) = \frac{1}{Z}p(\boldsymbol{x}_{t+1:T})p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})h_t(\boldsymbol{x}_t) = \tilde{p}(\boldsymbol{x}_{t+1:T})p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})\frac{h_t(\boldsymbol{x}_t)}{h_{t+1}(\boldsymbol{x}_{t+1})}.$$

This motivates defining the tilted transition kernel

$$\tilde{p}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}) = p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})\frac{h_t(\boldsymbol{x}_t)}{h_{t+1}(\boldsymbol{x}_{t+1})}, \tag{5}$$

under which $\tilde{p}(\boldsymbol{x}_{t:T}) = \tilde{p}(\boldsymbol{x}_{t+1:T})\tilde{p}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$. At time $t = T$, the corresponding marginal is

$$\tilde{p}(\boldsymbol{x}_T) = \frac{1}{Z}p(\boldsymbol{x}_T)h_T(\boldsymbol{x}_T)$$

Remark 3.1. *Using the difference potential $G_t(\boldsymbol{x}_{t:T}) = h_t(\boldsymbol{x}_t)/h_{t+1}(\boldsymbol{x}_{t+1})$ within the FK Steering framework yields the same reward-tilted target distribution.*

3.2 Residual Decomposition and pCN Steering

We now construct a practical sampling procedure for the locally tilted transition kernels in equation 5. Unlike FK Steering, which approximates a tilted path measure via SMC, our framework applies Metropolis–Hastings corrections to diffusion proposals, directly targeting the reward-tilted marginals without the mode collapse that can arise from particle resampling.

Recall that each reverse transition kernel is Gaussian: $p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}) = \mathcal{N}(\boldsymbol{x}_t;\, \boldsymbol{\mu}(\boldsymbol{x}_{t+1}, t+1), \tau_{t+1}^2\boldsymbol{I})$, where $\tau_{t+1} > 0$ is the standard deviation of the reverse kernel. Therefore, a sample from the reverse transition $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})$ can be written as

$$\boldsymbol{x}_t = \boldsymbol{\mu}(\boldsymbol{x}_{t+1}, t+1) + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \tau_{t+1}^2\boldsymbol{I}), \tag{6}$$

This decomposition exposes a Gaussian reference measure over the residual noise $\boldsymbol{\epsilon}$. Under this parameterization, sampling from the tilted transition $\tilde{p}(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})$ is equivalent to sampling residuals from the tilted residual distribution

$$\pi_t(\boldsymbol{\epsilon}|\boldsymbol{x}_{t+1}) \propto \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \tau_{t+1}^2\boldsymbol{I})h_t(\boldsymbol{\mu}(\boldsymbol{x}_{t+1}, t+1) + \boldsymbol{\epsilon})$$

We sample from $\pi_t(\boldsymbol{\epsilon}|\boldsymbol{x}_{t+1})$ using the pCN algorithm, which preserves the Gaussian reference distribution $\mathcal{N}(\mathbf{0}, \tau_{t+1}^2\boldsymbol{I})$. Concretely, given the current residual $\boldsymbol{\epsilon}$, we compute the pCN proposal

$$\hat{\boldsymbol{\epsilon}} = \sqrt{1-\lambda^2}\,\boldsymbol{\epsilon} + \lambda\boldsymbol{\xi}, \qquad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \tau_{t+1}^2\boldsymbol{I}), \tag{7}$$

and define the corresponding state $\hat{\boldsymbol{x}}_t = \boldsymbol{\mu}(\boldsymbol{x}_{t+1}, t+1) + \hat{\boldsymbol{\epsilon}}$. Since the pCN proposal preserves the Gaussian reference measure and is reversible, the acceptance probability depends solely on the intermediate potential:

$$\alpha(\boldsymbol{\epsilon}, \hat{\boldsymbol{\epsilon}}) = \min\left\{1, \frac{h_t(\hat{\boldsymbol{x}}_t)}{h_t(\boldsymbol{x}_t)}\right\} \tag{8}$$

While pCN provides an exact correction for the locally tilted transition, efficient exploration can remain challenging when the target is highly concentrated or multimodal. This motivates an additional exploration mechanism discussed in the next section.

Algorithm 1 RpCN

Require: replica count $K+1$, diffusion steps T, intermediate potentials $\{\bar{h}_t^\theta(\boldsymbol{x}_t;\beta_k)\}_{k=0}^K$, pCN step sizes $\{\lambda_k\}_{k=0}^K$, pCN updates M, reverse mean $\boldsymbol{\mu}_\theta(\boldsymbol{x}_{t+1}, t+1)$

for $t = T, \dots, 0$ **do**
  for $k = 0, \dots, K$ **do** ▷ Initialize replicas $\boldsymbol{x}_t^{(k)}$
    if $t = T$ **then**
      $\boldsymbol{\mu}_\theta^{(k)} \leftarrow \mathbf{0}, \quad \boldsymbol{\epsilon}^{(k)} \sim \mathcal{N}(\mathbf{0}, \sigma_T^2 \boldsymbol{I})$
    else
      $\boldsymbol{\mu}_\theta^{(k)} \leftarrow \boldsymbol{\mu}_\theta(\boldsymbol{x}_{t+1}^{(k)}, t+1), \quad \boldsymbol{\epsilon}^{(k)} \sim \mathcal{N}(\mathbf{0}, \tau_{t+1}^2 \boldsymbol{I})$
    end if
  end for
  if steering is applied at time step t **then**
    for $m = 1, \dots, M$ **do**
      for $k = 0, \dots, K$ **do** ▷ pCN update
        $\hat{\boldsymbol{\epsilon}}^{(k)} \leftarrow \sqrt{1-\lambda_k^2}\,\boldsymbol{\epsilon}^{(k)} + \lambda_k \boldsymbol{\xi}^{(k)}, \boldsymbol{\xi}^{(k)} \sim \mathcal{N}(\mathbf{0}, \tau_{t+1}^2 \boldsymbol{I})$
        Accept $\hat{\boldsymbol{\epsilon}}^{(k)}$ for $\boldsymbol{\epsilon}^{(k)}$ with probability α (equation 8) using $\bar{h}_t^\theta(\,\cdot\,;\beta_k)$.
      end for
      for $k = 0, \dots, K-1$ **do** ▷ Replica exchange
        Swap $(\boldsymbol{\mu}_\theta^{(k)}, \boldsymbol{\epsilon}^{(k)})$ with $(\boldsymbol{\mu}_\theta^{(k+1)}, \boldsymbol{\epsilon}^{(k+1)})$ with probability α_{swap} (equation 9) using $\bar{h}_t^\theta(\,\cdot\,;\beta_k)$ and $\bar{h}_t^\theta(\,\cdot\,;\beta_{k+1})$.
      end for
    end for
  end if
end for
$\boldsymbol{x}_0^{(K)} \leftarrow \boldsymbol{\mu}_\theta^{(K)} + \boldsymbol{\epsilon}^{(K)}$
return $\boldsymbol{x}_0^{(K)}$ ▷ Sample from $\tilde{p}(\boldsymbol{x}_0)$

3.3 Parallel Tempering

To improve exploration of multimodal reward landscapes, we integrate our method with parallel tempering (PT), also known as replica exchange [11, 44]. We maintain a population of $K+1$ replicas with corresponding inverse temperatures $0 = \beta_0 < \cdots < \beta_K = \beta$.

For each replica k, define the intermediate potential $h_t(\boldsymbol{x}_t;\beta_k) = \mathbb{E}_{p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\exp(\beta_k R(\boldsymbol{x}_0))]$. The trajectory of replica k, $\boldsymbol{x}_{t:T}^{(k)}$ targets the locally tilted distribution

$$p^{(k)}(\boldsymbol{x}_{t:T}^{(k)}) \propto p(\boldsymbol{x}_{t:T}^{(k)}) h_t(\boldsymbol{x}_t^{(k)};\beta_k).$$

After applying pCN updates to each replica in residual space (Section 3.2), PT proposes exchanges between adjacent replicas with probability

$$\alpha_{\text{swap}} = \min\left\{1, \frac{h_t(\boldsymbol{x}_t^{(k+1)};\beta_k) h_t(\boldsymbol{x}_t^{(k)};\beta_{k+1})}{h_t(\boldsymbol{x}_t^{(k)};\beta_k) h_t(\boldsymbol{x}_t^{(k+1)};\beta_{k+1})}\right\}. \tag{9}$$

These exchanges couple exploitation and exploration across the ladder of inverse temperatures. Replicas with large β_k concentrate on high-reward regions, while those with small β_k replicas explore flatter reward landscapes and maintain broader coverage of the state space. Replica exchanges allow the final replica assigned β_K to retain high-reward samples while benefiting from the continued exploration of the replicas with small β_k, thereby improving global exploration without altering the target distribution of the final replica assigned β_K.

3.4 RpCN Algorithm

Combining pCN with PT enables effective exploration of the sequence of locally tilted transition kernels induced by the intermediate rewards. Given a pretrained diffusion model $\boldsymbol{\mu}_\theta(\boldsymbol{x}_{t+1}, t+1)$ and a reward function $R(\boldsymbol{x})$, the resulting RpCN algorithm produces samples that approximate the

globally tilted target distribution $\tilde{p}_0(\boldsymbol{x}_0)$. We derived RpCN using the true reverse transition kernel $p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$, but in practice, we only have access to a learned approximation $p^\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$. This forces $h_t(\boldsymbol{x}_t;\beta_k)$ to be approximated by

$$h_t^\theta(\boldsymbol{x}_t;\beta_k) = \mathbb{E}_{p^\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\exp(\beta_k R(\boldsymbol{x}_0))].$$

The expectation in $h_t^\theta(\boldsymbol{x}_t;\beta_k)$ is still computationally intractable, so it must also approximated by some function $\bar{h}_t^\theta(\boldsymbol{x}_t;\beta_k)$. A common approach adopted in prior works (cf. [1, 6, 34]) use $\bar{h}_t^\theta(\boldsymbol{x}_t;\beta_k) = \exp(\beta_k R(\mathbb{E}_{p^\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\boldsymbol{x}_0]))$, where $\mathbb{E}_{p^\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\boldsymbol{x}_0]$ is the posterior mean computable using Tweedie's formula [12]. This approximation simplifies the acceptance probabilities where

$$\alpha(\boldsymbol{\epsilon},\hat{\boldsymbol{\epsilon}}) \approx \min\left\{1, \exp\left(\beta_k[R(\mathbb{E}_{p^\theta(\boldsymbol{x}_0|\hat{\boldsymbol{x}}_t)}[\boldsymbol{x}_0]) - R(\mathbb{E}_{p^\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\boldsymbol{x}_0])]\right)\right\},$$

$$\alpha_{\text{swap}} \approx \min\left\{1, \exp\left((\beta_{k+1}-\beta_k)\left[R(\mathbb{E}_{p^\theta(\boldsymbol{x}_0|\boldsymbol{x}_t^{(k)})}[\boldsymbol{x}_0]) - R(\mathbb{E}_{p^\theta(\boldsymbol{x}_0|\boldsymbol{x}_t^{(k+1)})}[\boldsymbol{x}_0])\right]\right)\right\}.$$

Here the number of function evaluations required to compute α_{swap} reduces from at least four to two. The complete RpCN procedure is given in Algorithm 1, and the theoretical impact of these approximations is studied in Section 4.

4 Theoretical Analysis

In this section, we present a theorem to clarify how the approximations made in Section 3.4 affect RpCN's accuracy in sampling the desired reward-tilted target distribution. The theorem requires two assumptions. First, to isolate the approximations' effects, we assume idealized local pCN mixing, meaning that each local sampler draws exactly from the tilted transition kernel $\tilde{p}^\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}) \propto p^\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})\bar{h}_t^\theta(\boldsymbol{x}_t)$ where $\bar{h}_t^\theta(\boldsymbol{x}_t)$ is the computationally tractable approximation of $h_t^\theta(\boldsymbol{x}_t)$. Second, we assume that, for all t, the range of the reward $R(\boldsymbol{x})$ is in $[R_{\min}, R_{\max}]$ and the range of $\log \bar{h}_t^\theta(\boldsymbol{x}_t)$ is in $[\beta R_{\min}, \beta R_{\max}]$. The theorem shows that the discrepancy between the marginals $\tilde{p}_0(\boldsymbol{x}_0)$ and $\tilde{p}_0^\theta(\boldsymbol{x}_0)$ is controlled by the approximations' errors accumulated along the reverse process. See Appendix A for the complete proof.

Theorem 4.1. *Suppose that, at each time step t, δ_t^2 bounds the expected squared L^2 error of the learned score and that η_t^2 bounds the expected squared error of the learned potential's approximation:*

$$\mathbb{E}_{p_t(\boldsymbol{x}_t)}[\|\boldsymbol{s}_\theta(\boldsymbol{x}_t,t) - \nabla \log p_t(\boldsymbol{x}_t,t)\|^2] \le \delta_t^2, \quad \mathbb{E}_{p_t(\boldsymbol{x}_t)}[(\bar{h}_t^\theta(\boldsymbol{x}_t) - h_t^\theta(\boldsymbol{x}_t))^2] \le \eta_t^2.$$

Let $B = \beta(R_{\max} - R_{\min})$. Under Assumptions A.1 and A.2, the KL divergence between the true tilted distribution $\tilde{p}_0(\boldsymbol{x}_0)$ and the generated tilted distribution $\tilde{p}_0^\theta(\boldsymbol{x}_0)$ is bounded:

$$D_{\text{KL}}\big(\tilde{p}_0(\boldsymbol{x}_0) \,\|\, \tilde{p}_0^\theta(\boldsymbol{x}_0)\big) \;\le\; C_0 \sum_{t=1}^{T} \delta_t^2 + C_1 \sum_{t=0}^{T} \eta_t + C_2 \sum_{t=1}^{T} \delta_t,$$

with constant coefficients $C_0 = e^B \max_{1\le t\le T} \frac{g_t^2}{2\tau_t^2}$, $C_1 = e^{-\beta R_{\min}}\Big(e^B + e^{2B}\frac{e^B-1}{B}\Big)$, and $C_2 = e^{2B}(e^B-1)\max_{1\le t\le T}\frac{|g_t|}{2\tau_t}$.

When the learned score is accurate along the reverse process and $\bar{h}_t^\theta(\boldsymbol{x}_t)$ approximates $h_t^\theta(\boldsymbol{x}_t)$ well, Theorem 4.1 shows that the final generated marginal is close to the target tilted distribution. Notably, as the learned score and learned potential's approximation improve ($\delta_t \to 0$ and $\eta_t \to 0$, respectively), the upper bound converges to zero, guaranteeing exact recovery of the tilted target distribution.

Remark 4.2. *Theorem 4.1 extends to equivalent loss functions, e.g., denoising loss [19, 42].*

5 Numerical Experiments

In this section, we empirically validate RpCN for reward-guided generation on a synthetic task and several benchmark tasks, including conditional image synthesis, dynamical system trajectory sampling, and noiseless inverse problem solving. We compare against SMC methods FK Steering [40] and DAS [29], and Parallel Tempering (PT) baseline CREPE [17]. Our comparisons with FK steering use the intermediate reward $R_t(\boldsymbol{x}_t) = R(\mathbb{E}_{p^\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\boldsymbol{x}_0])$. Full experimental details are provided in Appendix B.

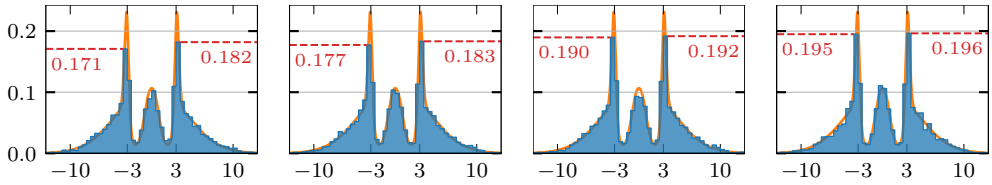


Figure 3: Comparison of empirical densities (blue histogram) of FK Steering, DAS, CREPE, and RpCN with 2,000 function evaluations with the true tilted density (orange line). The height of the two highest peaks of each empirical density is annotated (red lines). The KL divergence has unit 10^{-2}. All methods approximate the true tilted density well, but RpCN has the lowest KL divergence.

Model	Method	ImageReward ↑	CLIP Div ↑
Stable Diffusion 1.4	Base [38]	0.234	0.316
	DDPO [4]	0.263	–
	Best-of-N [46]	0.800	–
	FK Steering [40]	0.927	0.091
	DAS [29] ($\alpha_{\mathrm{KL}} = 0.1$)	0.672	0.025
	DAS [29] ($\alpha_{\mathrm{KL}} = 0.01$)	*1.257	0.013
	CREPE [17]	<u>1.001</u>	<u>0.331</u>
	RpCN **(ours)**	**1.054**	**0.337**
Stable Diffusion 1.5	Base [38]	0.187	0.312
	DPO[46]	0.343	–
	Best-of-N [46]	0.737	–
	FK Steering [40]	0.898	0.104
	DAS [29] ($\alpha_{\mathrm{KL}} = 0.1$)	0.686	0.027
	DAS [29] ($\alpha_{\mathrm{KL}} = 0.01$)	*1.144	0.008
	CREPE [17]	<u>1.022</u>	**0.340**
	RpCN **(ours)**	**1.046**	<u>0.338</u>
Stable Diffusion 2.1	Base [38]	0.372	0.248
	Best-of-N [46]	0.888	–
	FK Steering [40]	1.006	0.087
	CREPE [17]	**1.101**	**0.325**
	RpCN **(ours)**	<u>1.085</u>	<u>0.319</u>

Table 1: Average best-sample ImageReward and CLIP Diversity over ImageReward prompts, comparing RpCN against FK Steering and CREPE. Base, DDPO, DPO, and FK Steering results are from [40].
*DAS with $\alpha_{\mathrm{KL}} = 0.01$ corresponds to $\beta = 100$, but all other methods use $\beta = 10$; therefore, its results are not directly comparable.

5.1 Synthetic Experiment

We first benchmark RpCN on a simple 1D task with a tractable analytical solution. With a Gaussian prior distribution and a reward defined as a sum of exponentials, we show that RpCN is capable of sampling a multimodal tilted distribution, even when the its modes are not aligned with the modes of the prior distribution and are separated by low-density barriers. In the experiment, we train a diffusion model transporting $\mathcal{N}(0, 1)$ to $\mathcal{N}(0, 4^2)$, a prior distribution with a single mode at $x = 0$. The tilted target distribution is defined using the reward $R(x) = r(x) + r(-x)$ where $r(x) \propto 1.6615\mathcal{N}(x; 3, (0.5)^2) - 2\mathcal{N}(x; 2, 1)$ and tilt strength $\beta = 1$. It has two high-density modes around $x \in \{-3, 3\}$ separated by low-density barriers around $x \in \{-2, 2\}$. Fig. 3 compares the empirical densities of RpCN, FK Steering, DAS, and CREPE. While each method appears to effectively sample from the tilted target, all the methods struggle to approximate the high-density modes around $x \in \{-3, 3\}$. RpCN's empirical density, however, best approximates the density of these modes. The distributional alignment of RpCN is also reflected in it attaining the lowest KL divergence. Overall, PT and pCN enhance exploration, enabling sampling from high-density modes even across low-density barriers.

5.2 Conditional Image Generation

We evaluate RpCN on steering Stable Diffusion [38] toward high-quality, preference-aligned image generation, using the preference-aligned ImageReward model [50] as our reward function. We

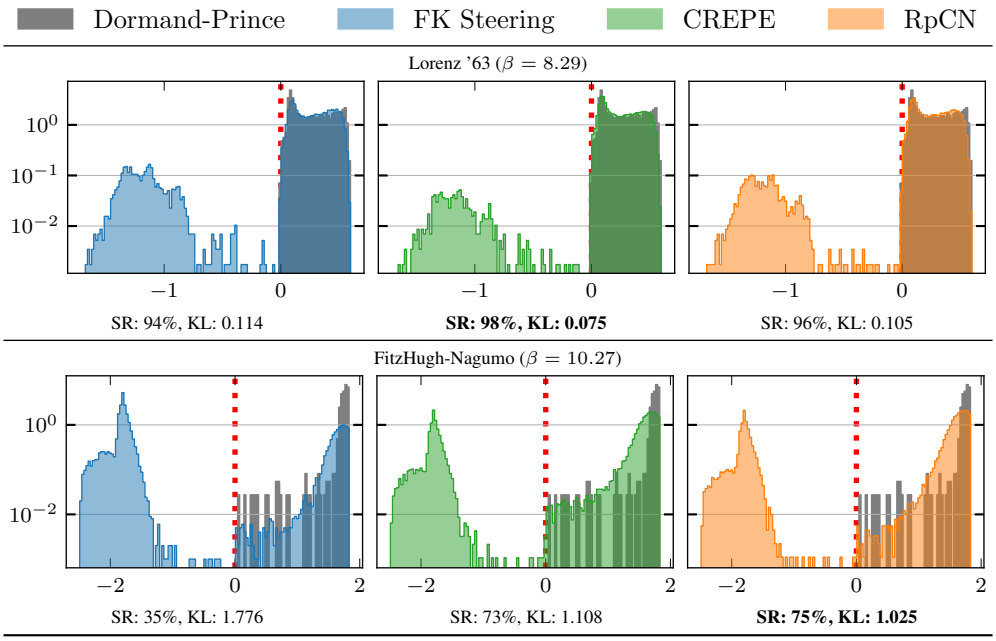


Figure 4: Comparison of $C(\boldsymbol{x})$ histograms where $\boldsymbol{x}$ is is a sampled trajectory. We also report the success rate (SR) and KL divergence from the ground truth Dormand-Prince [9] histogram (gray).

compare all methods with $\beta = 10$, which corresponds to $\alpha_{\text{KL}} = 0.1$ for DAS. We also report DAS with $\alpha_{\text{KL}} = 0.01$ ($\beta = 100$) to show its improvement in ImageReward with stronger tilt; however, it samples a different target tilted distribution so this result is not directly comparable.

Table 1 compares the best-sample ImageReward and CLIP diversity of the coldest replica, averaged over prompts. The SMC methods FK-Steering and DAS generate high-reward samples but exhibit a substantial reduction in sample diversity. In contrast, the parallel tempering methods RpCN and CREPE demonstrate strong reward alignment while preserving high sample diversity. Compared to FK Steering, RpCN consistently improves ImageReward while achieving over $3\times$ the sample diversity across all tested Stable Diffusion models. Relative to CREPE, RpCN improves both metrics on Stable Diffusion 1.4 and achieves higher ImageReward with comparable diversity on Stable Diffusion 1.5. On Stable Diffusion 2.1, CREPE achieves slightly higher values in both metrics. Overall, these results demonstrate that RpCN provides a favorable tradeoff between reward alignment and sample diversity, avoiding the sample diversity collapse exhibited by the SMC methods.

5.3 Dynamical Systems

We apply FK Steering, CREPE, and RpCN to tilt a diffusion model trained on dynamical system trajectories to sample trajectories $\boldsymbol{x}$ in an event set E [14] where $\boldsymbol{x} \in E$ if $C(\boldsymbol{x}) > 0$ for some constraint function C. Since set inclusion is a binary condition, we assume $p(\boldsymbol{x}|E) \propto p(\boldsymbol{x}) \exp(\beta R(\boldsymbol{x}))$ and choose $R(\boldsymbol{x}) = \text{sign}(C(\boldsymbol{x}))$, a discontinuous reward. Following [14], the two dynamical systems and events considered are Lorenz '63 [33], with the event of $\boldsymbol{x}$ saying on one arm the Lorenz attractor, and FitzHugh-Nagumo [15, 36], with the event of neuron spiking. Empirically, the events of Lorenz '63 and FitzHugh-Nagumo have success rates $p(E) \approx 1/5$ and $p(E) \approx 1/30$, respectively [14, 21, 22]. Precise characterizations of these events are given in Appendix B.2, as well as the grid search used to tune the methods' hyperparameters. Fig. 4 compares the histograms of $C(\boldsymbol{x})$ where $\boldsymbol{x}$ is sampled using each method; rejection sampling using Dormand-Prince [9] gives the reference histogram. All methods noticeably increase the success rate. CREPE is best on Lorenz '63, with a 98% success rate and 0.075 KL divergence versus RpCN with 96% and 0.105. On FitzHugh-Nagumo, RpCN is best with a 75% success rate and 1.025 KL divergence versus CREPE with 73% and 1.108. For both systems, there is a 2% difference in success rate between RpCN and CREPE, but RpCN is best with the rarer event.



(a) Ground Truth (b) Observation (c) RpCN (Ours) PSNR: 33.50 (d) DAS PSNR: 33.09 (e) DAPS PSNR: 31.53

Figure 5: Contrasting reconstruction quality between RpCN, DAS, and DAPS on the task of noiseless random inpainting. PSNR values are reported beneath each reconstruction.

5.4 Inverse Problems

We consider a variety of noiseless image restoration inverse problems, where balancing distributional fidelity and reconstruction accuracy is critical. We evaluate four operators—super-resolution, Gaussian deblurring, motion deblurring, and random inpainting—on 100 samples from the FFHQ [27] validation set, evaluating samples with PSNR and SSIM for reconstruction fidelity, LPIPS for perceptual quality, and FID for distributional fidelity.

Table 2 compares average PSNR, SSIM, and LPIPS across the 100 images, as well as the FID between the reconstructed and ground truth samples. We observe that RpCN achieves the best performance on 11 of 16 operator–metric pairs and ranks among the top two methods on all 16. Averaged across all four operators, RpCN improves PSNR and SSIM over DAPS by 3.8% and 6.5%, respectively, while reducing LPIPS by 52.5% and FID by 47.9%. Notably, on random inpainting, RpCN outperforms both DAS and DAPS across all four metrics, demonstrating that perceptual improvements need not come at the expense of reconstruction fidelity. On motion deblurring, DAS achieves slightly better structural and perceptual metrics, while RpCN attains the highest PSNR. Qualitatively, Fig. 5 demonstrates RpCN's ability to recover fine details such as hair texture that DAPS misses. Overall, these results indicate that RpCN achieves a favorable balance between distributional fidelity and perceptual quality, consistently delivering competitive or superior performance across all four inverse problems.

6 Conclusion

We introduced RpCN, a new test-time guidance method for diffusion models. By applying pCN updates in the residual space of each reverse diffusion step and integrating PT, RpCN enables robust exploration of multimodal reward landscapes, addressing a key limitation of SMC methods such as FK Steering. We further provided a theoretical KL-divergence bound to the reward-tilted target in terms of the score-matching error and learned potential approximation error. Empirically, RpCN is competitive with or outperforms baseline methods across a variety of tasks.

Operator	Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Super Resolution (×4)	RpCN (Ours)	<u>29.98</u>	**0.852**	**0.080**	**25.44**
	DAS [29]	26.19	0.746	<u>0.156</u>	53.01
	DAPS [51]	**30.36**	<u>0.812</u>	0.162	<u>48.62</u>
Inpaint (Random)	RpCN (Ours)	**34.94**	**0.950**	**0.032**	**10.54**
	DAS [29]	<u>33.86</u>	<u>0.940</u>	<u>0.038</u>	<u>16.75</u>
	DAPS [51]	32.00	0.869	0.119	29.70
Deblur (Gaussian)	RpCN (Ours)	<u>29.76</u>	**0.840**	**0.100**	**32.68**
	DAS [29]	28.45	<u>0.810</u>	<u>0.122</u>	<u>42.73</u>
	DAPS [51]	**30.19**	0.804	0.164	51.54
Deblur (Motion)	RpCN (Ours)	**35.43**	<u>0.937</u>	<u>0.050</u>	<u>15.81</u>
	DAS [29]	<u>35.24</u>	**0.943**	**0.045**	**13.65**
	DAPS [51]	32.82	0.874	0.107	32.39

Table 2: Truncated comparison of inverse problem solving results on FFHQ 256×256. Best results are in **bold**; second-best results are <u>underlined</u>.

Acknowledgments

This material is based on research sponsored by NSF grants DMS-2208361, DMS-2219956, DMS-2436344, and DMS-2608502, and DOE grants DE-SC0023490, DE-SC0025589, and DE-SC0025801.

Broader Impacts

RpCN introduces a flexible guidance framework for steering diffusion models, which carries the inherent risk of dual-use if paired with malicious reward functions. However, this potential for misuse is outweighed by the utility of the framework in a scientific discovery context. Our work on inverse problems shows that RpCN effectively balances the trade-off between distributional fidelity and reward alignment. By enforcing this balance, RpCN enables researchers to guide generative models toward samples with therapeutic properties without sacrificing domain-specific constraints. This capability is essential for generating viable candidates in fields such as in-silico drug design, where structures must be both functionally optimized and chemically realistic.

References

[1] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 843–852, 2023.

[2] Feng Bao, Zezhong Zhang, and Guannan Zhang. An ensemble score filter for tracking high-dimensional nonlinear dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 432:117447, 2024.

[3] Alexandros Beskos, Gareth Roberts, Andrew Stuart, and Jochen Voss. Mcmc methods for diffusion bridges. *Stochastics and Dynamics*, 8(03):319–350, 2008.

[4] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning, 2024.

[5] Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020.

[6] Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems, 2024.

[7] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. Mcmc methods for functions: Modifying old algorithms to make them faster. *Statistical Science*, 28(3), August 2013.

[8] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[9] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

[10] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[11] David J Earl and Michael W Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

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

[13] Mohammad Farazmand and Themistoklis P Sapsis. Extreme events: Mechanisms and prediction. *Applied Mechanics Reviews*, 71(5):050801, 2019.

[14] Marc Anton Finzi, Anudhyan Boral, Andrew Gordon Wilson, Fei Sha, and Leonardo Zepeda-Nunez. User-defined event sampling and uncertainty quantification in diffusion models for physical dynamical systems. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10136–10152. PMLR, 23–29 Jul 2023.

[15] Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466, 1961.

[16] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[17] Jiajun He, Paul Jeha, Peter Potaptchik, Leo Zhang, José Miguel Hernández-Lobato, Yuanqi Du, Saifuddin Syed, and Francisco Vargas. Crepe: Controlling diffusion with replica exchange. *arXiv preprint arXiv:2509.23265*, 2025.

[18] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J. Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold preserving guided diffusion, 2023.

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

[20] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[21] Yuhao Huang, Taos Transue, Shih-Hsin Wang, William M Feldman, Hong Zhang, and Bao Wang. Improving flow matching by aligning flow divergence. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 25813–25834. PMLR, 13–19 Jul 2025.

[22] Yuhao Huang, Shih-Hsin Wang, Andrea L. Bertozzi, and Bao Wang. RMFlow: Refined mean flow by a noise-injection step for multimodal generation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[23] Aapo Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

[24] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[25] Jérôme Idier. *Bayesian approach to inverse problems*. John Wiley & Sons, 2013.

[26] 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.

[27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[28] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models, 2022.

[29] Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[30] Tomasz Korbak, Ethan Perez, and Christopher Buckley. Rl with kl penalties is better viewed as bayesian inference. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1083–1091, 2022.

[31] Xiang Li, Soo Min Kwon, Shijun Liang, Ismail R. Alkhouri, Saiprasad Ravishankar, and Qing Qu. Decoupled data consistency with diffusion purification for image restoration, 2025.

[32] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

[34] Konstantin Mark, Leonard Galustian, Maximilian P-P Kovar, and Esther Heid. Feynman-kac-flow: Inference steering of conditional flow matching to an energy-tilted posterior. *arXiv preprint arXiv:2509.01543*, 2025.

[35] Christian A Naesseth, Fredrik Lindsten, and Thomas B Schön. Elements of sequential monte carlo. *Foundations and Trends® in Machine Learning*, 12(3):187–306, 2019.

[36] J Nagumo, S Arimoto, and S Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962.

[37] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

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

[39] Christopher Scarvelis, Haitz Sáez de Ocáriz Borde, and Justin Solomon. Closed-form diffusion models. *Transactions on Machine Learning Research*, 2025.

[40] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

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

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

[43] Andrew M Stuart. Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559, 2010.

[44] Robert H. Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Phys. Rev. Lett.*, 57:2607–2609, Nov 1986.

[45] Taos Transue, Bohan Chen, So Takao, and Bao Wang. Flow matching for efficient and scalable data assimilation. *arXiv preprint arXiv:2508.13313*, 2025.

[46] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization, 2023.

[47] Shih-Hsin Wang, Joel Keller, Taos Transue, Drake Benjamin Brown, Thomas Strohmer, and Bao Wang. Test-time guidance for flow-based generative models via parallel tempering on source distributions. In *Forty-third International Conference on Machine Learning*, 2026.

[48] Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model, 2022.

[49] Zifan Wang, Alice Harting, Matthieu Barreau, Michael M Zavlanos, and Karl H Johansson. Source-guided flow matching. *arXiv preprint arXiv:2508.14807*, 2025.

[50] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

[51] Bingliang Zhang, Wenda Chu, Julius Berner, Chenlin Meng, Anima Anandkumar, and Yang Song. Improving diffusion inverse problem solving with decoupled noise annealing. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 20895–20905. IEEE, 2025.

[52] Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising diffusion models for plug-and-play image restoration, 2023.

A Technical Proofs

A.1 Proof of Theorem 4.1

We first summarize the assumptions that we impose.

Assumption A.1 (Idealized pCN mixing)**.** Let $\bar{h}_t^\theta(\boldsymbol{x}_t)$ denote the potential that RpCN evaluates in place of $h_t(\boldsymbol{x}_t)$, such as the Tweedie estimate of Section 3.4. Steering is applied at every $t \in \{T, \dots, 0\}$, and at each t the pCN MCMC chain has reached stationarity. Hence $\boldsymbol{x}_t$ is an exact draw from $\tilde{p}^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) \propto p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})\, \bar{h}_t^\theta(\boldsymbol{x}_t)$ (from $\tilde{p}_T^\theta(\boldsymbol{x}_T) \propto p_T(\boldsymbol{x}_T)\bar{h}_T^\theta(\boldsymbol{x}_T)$ at $t = T$).

Assumption A.2 (Boundedness)**.** For all $t \in \{T \dots, 0\}$, the range of $R(\boldsymbol{x})$ is in $[R_{\min}, R_{\max}]$ and the range of $\log \bar{h}_t^\theta(\boldsymbol{x})$ is in $[\beta R_{\min}, \beta R_{\max}]$. Note that the common definition of $\bar{h}_t^\theta(\boldsymbol{x}_t)$ discussed in Section 3.4 satisfies this assumption when R does.

To analyze the convergence of the terminal distributions, we lift the problem to the space of trajectory measures and apply the Data Processing Inequality (DPI), which bounds the KL divergence between the final tilted distributions by the KL divergence between their corresponding path measures. This enables a decomposition of the global error into a sum of local discrepancies between tilted transition kernels along the sampling trajectory. We then relate these kernel-level discrepancies to the score function, noting that each transition kernel depends on the learned score through the reverse-process mean and the Tweedie denoising estimator, and show that these local errors can be explicitly controlled by the score approximation error.

Theorem 4.1. *Suppose that, at each time step t, δ_t^2 bounds the expected squared L^2 error of the learned score and that η_t^2 bounds the expected squared error of the learned potential's approximation:*

$$\mathbb{E}_{p_t(\boldsymbol{x}_t)}[\|\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla \log p_t(\boldsymbol{x}_t, t)\|^2] \le \delta_t^2, \quad \mathbb{E}_{p_t(\boldsymbol{x}_t)}[(\bar{h}_t^\theta(\boldsymbol{x}_t) - h_t^\theta(\boldsymbol{x}_t))^2] \le \eta_t^2.$$

Let $B = \beta(R_{\max} - R_{\min})$. Under Assumptions A.1 and A.2, the KL divergence between the true tilted distribution $\tilde{p}_0(\boldsymbol{x}_0)$ and the generated tilted distribution $\tilde{p}_0^\theta(\boldsymbol{x}_0)$ is bounded:

$$D_{\mathrm{KL}}\big(\tilde{p}_0(\boldsymbol{x}_0) \,\|\, \tilde{p}_0^\theta(\boldsymbol{x}_0)\big) \;\le\; C_0 \sum_{t=1}^{T} \delta_t^2 + C_1 \sum_{t=0}^{T} \eta_t + C_2 \sum_{t=1}^{T} \delta_t,$$

with constant coefficients $C_0 = e^B \max_{1\le t\le T} \frac{g_t^2}{2\tau_t^2}$, $C_1 = e^{-\beta R_{\min}}\Big(e^B + e^{2B}\frac{e^B-1}{B}\Big)$, and $C_2 = e^{2B}(e^B - 1)\max_{1\le t\le T} \frac{|g_t|}{2\tau_t}$.

Proof of Theorem 4.1. Step 1: path distributions. Section 3.1 defines the true tilted path distribution as $\tilde{p}(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_{0:T})e^{\beta R(\boldsymbol{x}_0)}/Z$, and then derives the initial law $\tilde{p}_T(\boldsymbol{x}_T) = p_T(\boldsymbol{x}_T)h_T(\boldsymbol{x}_T)/Z$ and reverse transition kernels (see equation 5). Integrating $\tilde{p}(\boldsymbol{x}_{0:T})$ over $\boldsymbol{x}_{0:t-1}$ and $\boldsymbol{x}_{t+1:T}$ gives its marginals

$$\tilde{p}_t(\boldsymbol{x}_t) = \frac{p_t(\boldsymbol{x}_t)\, h_t(\boldsymbol{x}_t)}{Z}, \qquad t \in \{0, \dots, T\}. \tag{10}$$

The same derivation steps applied to the learned path $p^\theta(\boldsymbol{x}_{0:T}) = p_T(\boldsymbol{x}_T)\prod_{t=0}^{T-1} p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})$ shows that its tilted path distribution

$$\check{p}^\theta(\boldsymbol{x}_{0:T}) := \frac{p^\theta(\boldsymbol{x}_{0:T})\, e^{\beta R(\boldsymbol{x}_0)}}{Z^\theta}, \qquad Z^\theta = \mathbb{E}_{p_0^\theta(\boldsymbol{x}_0)}\big[e^{\beta R(\boldsymbol{x}_0)}\big],$$

has the initial law $\check{p}_T^\theta(\boldsymbol{x}_T) = p_T(\boldsymbol{x}_T)h_T^\theta(\boldsymbol{x}_T)/Z^\theta$ and kernels $\check{p}^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) = p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})\, h_t^\theta(\boldsymbol{x}_t)/h_{t+1}^\theta(\boldsymbol{x}_{t+1})$; in particular $\mathbb{E}_{p^\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}[h_t^\theta(\boldsymbol{x}_t)] = h_{t+1}^\theta(\boldsymbol{x}_{t+1})$. The path distribution $\check{p}^\theta(\boldsymbol{x}_{0:T})$ is what RpCN would generate with the exact potentials of the learned model $h_t^\theta(\boldsymbol{x}_t)$, but these are generally computationally intractable. Instead, RpCN approximates $h_t^\theta(\boldsymbol{x}_t)$ with $\bar{h}_t^\theta(\boldsymbol{x}_t)$. By Assumption A.1, RpCN generates the path distribution

$$\tilde{p}^\theta(\boldsymbol{x}_{0:T}) = \tilde{p}_T^\theta(\boldsymbol{x}_T) \prod_{t=0}^{T-1} \tilde{p}^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}),$$

$$\tilde{p}_T^\theta(\boldsymbol{x}_T) = \frac{p_T(\boldsymbol{x}_T)\bar{h}_T^\theta(\boldsymbol{x}_T)}{\bar{Z}}, \qquad \tilde{p}^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) = p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})\frac{\bar{h}_t^\theta(\boldsymbol{x}_t)}{\bar{N}_t(\boldsymbol{x}_{t+1})},$$

where $\bar{Z} = \mathbb{E}_{p_T(\boldsymbol{x}_T)}[\bar{h}_T^\theta(\boldsymbol{x}_T)]$ and $\bar{N}_t(\boldsymbol{x}_{t+1}) = \mathbb{E}_{p^\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}[\bar{h}_t^\theta(\boldsymbol{x}_t)]$.

We compare $\tilde{p}(\boldsymbol{x}_{0:T})$ with $\tilde{p}^\theta(\boldsymbol{x}_{0:T})$ through $\check{p}^\theta(\boldsymbol{x}_{0:T})$ by

$$D_{\mathrm{KL}}\big(\tilde{p}(\boldsymbol{x}_{0:T}) \,\|\, \tilde{p}^\theta(\boldsymbol{x}_{0:T})\big) = D_{\mathrm{KL}}\big(\tilde{p}(\boldsymbol{x}_{0:T}) \,\|\, \check{p}^\theta(\boldsymbol{x}_{0:T})\big) + \mathbb{E}_{\tilde{p}(\boldsymbol{x}_{0:T})}\Big[\log \frac{\check{p}^\theta(\boldsymbol{x}_{0:T})}{\tilde{p}^\theta(\boldsymbol{x}_{0:T})}\Big]. \tag{11}$$

The first term is due to the error of the learned kernels, and is bounded in step 2. The second term is due to the error of the potentials and is bounded in step 3.

Step 2: kernel error. Both $\tilde{p}(\boldsymbol{x}_{0:T})$ and $\check{p}^\theta(\boldsymbol{x}_{0:T})$ tilt a path distribution by the same function $e^{\beta R(\boldsymbol{x}_0)}$. Augment $\boldsymbol{x}_{0:T}$, under both $p(\boldsymbol{x}_{0:T})$ and $p^\theta(\boldsymbol{x}_{0:T})$, with a binary variable A satisfying

$$p(A = 1 \mid \boldsymbol{x}_{0:T}) = p^\theta(A = 1 \mid \boldsymbol{x}_{0:T}) = \frac{e^{\beta R(\boldsymbol{x}_0)}}{e^{\beta R_{\max}}} \in (0, 1].$$

Bayes' rule gives $p(\boldsymbol{x}_{0:T} \mid A = 1) = \tilde{p}(\boldsymbol{x}_{0:T})$ and $p^\theta(\boldsymbol{x}_{0:T} \mid A = 1) = \check{p}^\theta(\boldsymbol{x}_{0:T})$, and $p(A = 1) = Z/e^{\beta R_{\max}} \geq e^{-B}$. We apply the KL chain rule to the joint laws of $(\boldsymbol{x}_{0:T}, A)$. When conditioning first on $\boldsymbol{x}_{0:T}$, the conditional laws of A coincide, so

$$D_{\mathrm{KL}}\big(p(\boldsymbol{x}_{0:T}, A) \,\|\, p^\theta(\boldsymbol{x}_{0:T}, A)\big) = D_{\mathrm{KL}}\big(p(\boldsymbol{x}_{0:T}) \,\|\, p^\theta(\boldsymbol{x}_{0:T})\big).$$

On the other hand, when conditioning first on A, and dropping nonnegative terms,

$$D_{\mathrm{KL}}\big(p(\boldsymbol{x}_{0:T}, A) \,\|\, p^\theta(\boldsymbol{x}_{0:T}, A)\big) \geq p(A = 1)\, D_{\mathrm{KL}}\big(\tilde{p}(\boldsymbol{x}_{0:T}) \,\|\, \check{p}^\theta(\boldsymbol{x}_{0:T})\big).$$

Hence

$$D_{\mathrm{KL}}\big(\tilde{p}(\boldsymbol{x}_{0:T}) \,\|\, \check{p}^\theta(\boldsymbol{x}_{0:T})\big) \leq e^B\, D_{\mathrm{KL}}\big(p(\boldsymbol{x}_{0:T}) \,\|\, p^\theta(\boldsymbol{x}_{0:T})\big). \tag{12}$$

The path distributions $p(\boldsymbol{x}_{0:T})$ and $p^\theta(\boldsymbol{x}_{0:T})$ have the same initial law $p_T(\boldsymbol{x}_T)$, so their KL reduces to the KL between pairs of corresponding reverse transition kernels. Each pair of kernels are Gaussians with the same covariance $\tau_{t+1}^2 \boldsymbol{I}$, so

$$\begin{aligned} D_{\mathrm{KL}}\big(p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) \,\|\, p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})\big) &= \frac{\big\|\boldsymbol{\mu}_\theta(\boldsymbol{x}_{t+1}, t+1) - \boldsymbol{\mu}(\boldsymbol{x}_{t+1}, t+1)\big\|^2}{2\tau_{t+1}^2} \\ &= \frac{g_{t+1}^2 \|\boldsymbol{s}_\theta(\boldsymbol{x}_{t+1}, t+1) - \nabla \log p_{t+1}(\boldsymbol{x}_{t+1})\|^2}{2\tau_{t+1}^2} \end{aligned} \tag{13}$$

where the last equality used the relation in equation 3. By the hypothesis on $\boldsymbol{s}_\theta$,

$$D_{\mathrm{KL}}\big(p(\boldsymbol{x}_{0:T}) \,\|\, p^\theta(\boldsymbol{x}_{0:T})\big) = \sum_{t=0}^{T-1} \mathbb{E}_{p_{t+1}(\boldsymbol{x}_{t+1})}\Big[D_{\mathrm{KL}}\big(p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) \,\|\, p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})\big)\Big] \leq \sum_{t=1}^{T} \frac{g_t^2 \delta_t^2}{2\tau_t^2}, \tag{14}$$

and by equation 12 the first term of equation 11 is at most $e^B \sum_{t=1}^T g_t^2 \delta_t^2/(2\tau_t^2) \leq C_0 \sum_{t=1}^T \delta_t^2$ where $C_0 = e^B \max_{1 \leq t \leq T} g_t^2/(2\tau_t^2)$.

Step 3: potential error. Note the following facts:

(a) By convexity, $e^x - 1 \leq c_B x$ where $c_B = (e^B - 1)/B$ (with $c_0 = 0$) for $0 \leq x \leq B$. Also, $e^x - 1 < 0$ for $x < 0$. Therefore, $e^x - 1 \leq c_B |x|$ for all $x \leq B$. Together with $\log y \leq y - 1$, this gives $\log \mathbb{E}[e^\Delta] \leq c_B\, \mathbb{E}|\Delta|$ for any random variable $\Delta \leq B$.

(b) The potentials $h_t(\boldsymbol{x}_t)$, $h_t^\theta(\boldsymbol{x}_t)$ and $\bar{h}_t^\theta(\boldsymbol{x}_t)$ are all in $[e^{\beta R_{\min}}, e^{\beta R_{\max}}]$, and $Z, Z^\theta \geq e^{\beta R_{\min}}$. Equation 10 gives $\tilde{p}_t(\boldsymbol{x}_t) \leq e^B p_t(\boldsymbol{x}_t)$, and the definitions give $\check{p}_T^\theta(\boldsymbol{x}_T) \leq e^B p_T(\boldsymbol{x}_T)$ and $\check{p}^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) \leq e^B p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})$.

Let $\Delta_t(\boldsymbol{x}_t) = \log \bar{h}_t^\theta(\boldsymbol{x}_t) - \log h_t^\theta(\boldsymbol{x}_t)$, and note that $|\Delta_t(\boldsymbol{x}_t)| \leq B$. Since $|\log x - \log y| \leq |x - y|/e^{\beta R_{\min}}$ for $x, y \geq e^{\beta R_{\min}}$, by applying Jensen's inequality to $\sqrt{\cdot}$,

$$\mathbb{E}_{p_t(\boldsymbol{x}_t)}|\Delta_t(\boldsymbol{x}_t)| \leq e^{-\beta R_{\min}}\, \mathbb{E}_{p_t(\boldsymbol{x}_t)}\big|\bar{h}_t^\theta(\boldsymbol{x}_t) - h_t^\theta(\boldsymbol{x}_t)\big| \leq e^{-\beta R_{\min}} \eta_t. \tag{15}$$

Initial term. By the tower property for the learned path, $Z^\theta = \mathbb{E}_{p_T(\boldsymbol{x}_T)}[h_T^\theta(\boldsymbol{x}_T)]$, so $\bar{Z}/Z^\theta = \mathbb{E}_{\check{p}_T^\theta(\boldsymbol{x}_T)}[e^{\Delta_T(\boldsymbol{x}_T)}]$ and

$$\log \frac{\check{p}_T^\theta(\boldsymbol{x}_T)}{\tilde{p}_T^\theta(\boldsymbol{x}_T)} = -\Delta_T(\boldsymbol{x}_T) + \log \mathbb{E}_{\check{p}_T^\theta(\boldsymbol{x}'_T)}\big[e^{\Delta_T(\boldsymbol{x}'_T)}\big].$$

Under $\tilde{p}(\boldsymbol{x}_{0:T})$, $\boldsymbol{x}_T$ has law $\tilde{p}_T$. By (a), (b) and equation 15, the expectation of this log-ratio is at most

$$e^B \, \mathbb{E}_{p_T(\boldsymbol{x}_T)} |\Delta_T(\boldsymbol{x}_T)| + c_B \, \mathbb{E}_{\tilde{p}_T^\theta(\boldsymbol{x}_T)} |\Delta_T(\boldsymbol{x}_T)| \le (1 + c_B) e^B e^{-\beta R_{\min}} \eta_T \le C_1 \eta_T.$$

where $C_1 = e^{-\beta R_{\min}}(e^B + c_B e^{2B})$.

Transition terms. For $t < T$, since $\mathbb{E}_{p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})}[h_t^\theta(\boldsymbol{x}_t)] = h_{t+1}^\theta(\boldsymbol{x}_{t+1})$, we have

$$\frac{\bar{N}_t(\boldsymbol{x}_{t+1})}{h_{t+1}^\theta(\boldsymbol{x}_{t+1})} = \mathbb{E}_{\check{p}^\theta(\boldsymbol{x}_t' \mid \boldsymbol{x}_{t+1})}\big[e^{\Delta_t(\boldsymbol{x}_t')}\big],$$

and therefore

$$\log \frac{\check{p}^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})}{\tilde{p}^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})} = -\Delta_t(\boldsymbol{x}_t) + \log \mathbb{E}_{\tilde{p}^\theta(\boldsymbol{x}_t' \mid \boldsymbol{x}_{t+1})}\big[e^{\Delta_t(\boldsymbol{x}_t')}\big].$$

Under $\tilde{p}(\boldsymbol{x}_{0:T})$, $\boldsymbol{x}_t$ has law $\tilde{p}_t$ and $\boldsymbol{x}_{t+1}$ has law $\tilde{p}_{t+1}$. For the first part, (b) and equation 15 give $\mathbb{E}_{\tilde{p}_t(\boldsymbol{x}_t)}[-\Delta_t(\boldsymbol{x}_t)] \le e^B e^{-\beta R_{\min}} \eta_t$. For the second part, (a) gives the bound $c_B \, \mathbb{E}_{m_t(\boldsymbol{x}_t)} |\Delta_t(\boldsymbol{x}_t)|$, where $m_t(\boldsymbol{x}_t) = \int \tilde{p}_{t+1}(\boldsymbol{x}_{t+1}) \, \tilde{p}^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) \, \mathrm{d}\boldsymbol{x}_{t+1}$. By (b), $m_t(\boldsymbol{x}_t) \le e^{2B} q_t^\theta(\boldsymbol{x}_t)$, where

$$q_t^\theta(\boldsymbol{x}_t) = \int p_{t+1}(\boldsymbol{x}_{t+1}) \, p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}) \, \mathrm{d}\boldsymbol{x}_{t+1}$$

is the one-step prediction of the learned kernel from the reference marginal. The joint laws $p_{t+1}(\boldsymbol{x}_{t+1}) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})$ and $p_{t+1}(\boldsymbol{x}_{t+1}) p^\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})$ have $\boldsymbol{x}_t$-marginals p_t and q_t^θ, so data processing, equation 13 and Pinsker's inequality give $\mathrm{TV}(p_t, q_t^\theta) \le |g_{t+1}| \delta_{t+1} / (2\tau_{t+1})$. Since $0 \le |\Delta_t(\boldsymbol{x}_t)| \le B$,

$$\mathbb{E}_{m_t(\boldsymbol{x}_t)} |\Delta_t(\boldsymbol{x}_t)| \le e^{2B} \Big(\mathbb{E}_{p_t(\boldsymbol{x}_t)} |\Delta_t(\boldsymbol{x}_t)| + B \, \mathrm{TV}(p_t, q_t^\theta) \Big) \le e^{2B} \Big(e^{-\beta R_{\min}} \eta_t + \frac{B \, |g_{t+1}| \delta_{t+1}}{2\tau_{t+1}} \Big).$$

Each transition term is at most $C_1 \eta_t + C_2 \, \delta_{t+1}$ where $C_2 = c_B B e^{2B} \max_{1 \le t \le T} |g_t| / (2\tau_t)$.

Conclusion. Summing the initial and transition terms bounds the second term of equation 11 by

$$C_1 \sum_{t=0}^{T} \eta_t + C_2 \sum_{t=1}^{T} \delta_t.$$

Together with Step 2, this bounds $D_{\mathrm{KL}}(\tilde{p}(\boldsymbol{x}_{0:T}) \,\|\, \tilde{p}^\theta(\boldsymbol{x}_{0:T}))$, and data processing under $\boldsymbol{x}_{0:T} \mapsto \boldsymbol{x}_0$ proves the claim. □

B Experimental Details

We packaged our implementations of FK Steering, DAS, CREPE, and RpCN into a Python library that guarantees that the neural network is never evaluated twice with the same input. For each experiment, we approximately match the number of network function evaluations by deciding the parameters of each method as follows.

Let $u \in [0, 1]$ represent a continuous sampling index where $x_u = x_T$ at $u = 1$ and $x_u = x_0$ at $u = 0$. We first decide the discretization of $[0, 1]$, the interval of the continuous sampling index u, for each method. FK Steering, DAS, and RpCN partition $[0, 1]$ into $T + 1$ evenly spaced intervals. CREPE partitions $[0, 1]$ into $L - 1$ evenly spaced intervals, and places a replica at each interval boundary, resulting in L replicas. Between each replica, $\lfloor \frac{T+1}{L-1} \rfloor$ sampling steps are taken to generate proposals, so that the discretization of the u-interval is approximately the same between CREPE and the other methods. Each experiment chooses T and then the following parameters for each method.

- **FK Steering**: the particle count N, and the potential (sum, max, or difference).
- **DAS**: the particle count N.
- **CREPE**: the number of replicas L, the number of unadjusted Langevin steps M_{lgvn} with step size h_u, and number of blocks M_{blk}. A block is a procedure that first generates proposals for all the even-indexed replicas and attempts to swap, and then does the same for the odd-indexed replicas; in Algorithm 1 of [17], a block is one iteration of the for loop over PT iterations.
- **RpCN**: the number replicas $K + 1$ and the number of pCN steps M.

In all experiments, we use Tweedie's estimate to compute the intermediate reward/potential.

Computational Resources All experiments were run on NVIDIA RTX 3090 GPUs. The computational overhead of RpCN is primarily controlled by the number of pCN proposal steps, the number of parallel tempering replicas, and the number of diffusion time steps at which the local MCMC updates are applied. For each task, we provide the hyperparameters and configurations required to reproduce our main experimental results and estimated runtimes.

B.1 Synthetic Example Experimental Details

Diffusion Model The model is a Monte Carlo approximation of a denoiser model inspired by [2, 45, 39], which construct Monte Carlo approximations of the score or flow matching vector field. We use the EDM [26] (variance exploding) probability path with $\sigma_{\min} = 10^{-3}$, $\sigma_{\max} = 10$, and approximate the denoiser using $S = 9{,}984$ training pairs $(x_T^{(s)}, x_0^{(s)})$ sampled from the joint distribution $p(x_T, x_0) = p(x_T)p(x_0) = \mathcal{N}(0, \sigma_{\max}^2)\mathcal{N}(0, 4^2)$. Let $u \in [0, 1]$ represent a continuous sampling index where $x_u = x_T$ at $u = 1$ and $x_u = x_0$ at $u = 0$. Since $p_u(x_u) = \int p(x_u|x_0)p(x_0)\,\mathrm{d}x_0$ and

$$\mathbb{E}_{p(x_0)}[x_0|x_u] = \int x_0 p(x_0|x_u)\,\mathrm{d}x_0 = \int x_0 \frac{p(x_u|x_0)p(x_0)}{p_u(x_u)}\,\mathrm{d}x_0$$

then we define the predicted clean sample as

$$x_{\mathrm{pred}}(x_u, u) = \sum_{s=1}^{S} w^{(s)}(x_u, u)x_0^{(s)}, \quad w^{(s)}(x_u, u) = \frac{\mathcal{N}(x_u; x_0^{(s)}, \sigma(u)^2)}{\sum_{r=1}^{S} \mathcal{N}(x_u; x_0^{(r)}, \sigma(u)^2)}$$

where $\sigma(u) = (\sigma_{\min}^{1/\rho} + u(\sigma_{\max}^{1/\rho} - \sigma_{\min}^{1/\rho}))^\rho$ is the noise level and $\rho = 7$.

Constructing Empirical Densities We first compute the true tilted target distribution by computing the normalizing constant of $p(x)\exp(\beta R(x))$. The empirical densities for FK Steering, CREPE and RpCN are computed by drawing 9,984 samples using each method, and then constructing a histogram with 60 bins partitioning the interval $[-15, 15]$. FK Steering and DAS use adaptive resampling [40], requiring the recomputation each particle's potential every diffusion time step to decide whether to resample. Similarly, RpCN performs pCN updates and replica exchanges every diffusion time step.

Metrics From Empirical Densities The KL divergence is computed with midpoint rule integration where the midpoints are the centers of the empirical densities' histogram bins.

Method Hyperparameters In this comparison, each method uses 2,000 network function evaluations. We use $T + 1 = 50$. For FK Steering and DAS, we use $N = 40$ particles and adaptive resampling (ESS) [40]. FK Steering uses the difference potential, and DAS uses the annealed gradient guidance proposal with annealing schedule $\min\{1, (1 + \gamma)^t - 1\}$ where $\gamma = 0.008$. For CREPE, we choose $L = 5$ replicas, $M_{\mathrm{lgvn}} = 1$ unadjusted Langevin correction steps, and $M_{\mathrm{blk}} = 22$ blocks. The tilt for CREPE is annealed over u, where for a replica at u, the tilt strength is $\beta_u = (1 - u)\beta$. CREPE's unadjusted Langevin correction step has step size $h_u = 0.01\sigma(u)^2$ with reward gradient drift. CREPE's proposals are also reward gradient guided. Finally, for RpCN, we choose $K + 1 = 4$, $M = 9$, replica tilts $\beta_k = \beta(k/K)^2$, and pCN step sizes $\lambda_k = 0.5$. All methods use the Euler-Maruyama sampler.

Ablation of RpCN Hyperparameters Fig. 6 shows the KL divergence of the empirical density and mean reward of the samples for RpCN varying the number of replicas $K + 1$ and the number of pCN steps M with a maximum of 2,000 network function evaluations. RpCN's performance is largely influenced by PT. Without PT ($K + 1 = 1$), RpCN's KL divergence is multiple times larger than FK Steering; however, for $K + 1 \geq 2$, there is a pCN step count M for which RpCN's KL divergence is lower. When $K + 1 \geq 3$, increasing the pCN steps M further improves performance. With enough pCN steps, RpCN's KL divergence is lower than all compared methods.

Computational Resources On one GPU, FK Steering, DAS, CREPE, and RpCN take less than three minutes to generate $9{,}984$ samples. We also run RpCN with 20 sets of hyperparameters, so the total compute is less than one hour.

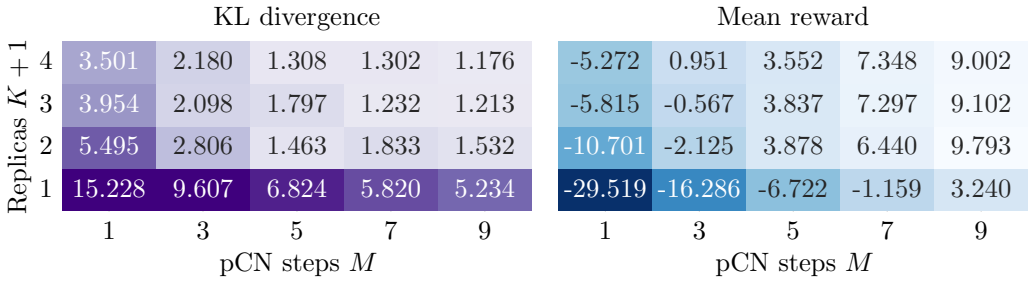


Figure 6: KL divergence and mean reward with unit 10^{-2} of the empirical densities of RpCN varying replica count $K+1$ and pCN steps M, using at most 2,000 function evaluations.

B.2 Dynamical Systems Trajectory Generation Experimental Details

The experimental setting follows [14, 21]. Each sampled trajectory is a discrete time series of states $\boldsymbol{x} = [\boldsymbol{x}(\tau_i)]_{i=1}^N \in \mathbb{R}^{Nd}$, which when conforming to some constraint (e.g., $C(\boldsymbol{x}) > 0$), is a trajectory of event E.

Trajectory Training Dataset The Dormand-Prince ODE solver [9] generates trajectories for training with absolute tolerance 1.4×10^{-8} and relative tolerance 1×10^{-6} for both dynamical systems with time step size 0.1 and 6.0 for Lorenz '63 and FitzHugh-Nagumo, respectively. For Lorenz '63, initial conditions are sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and evolved for 30 time steps for "burn-in." before being used. Similarly for FitzHugh-Nagumo, we sample initial conditions from $\mathcal{N}(\mathbf{0}, (0.2)^2\boldsymbol{I})$ with 250 burn-in time steps. All trajectories are 60 time steps long.

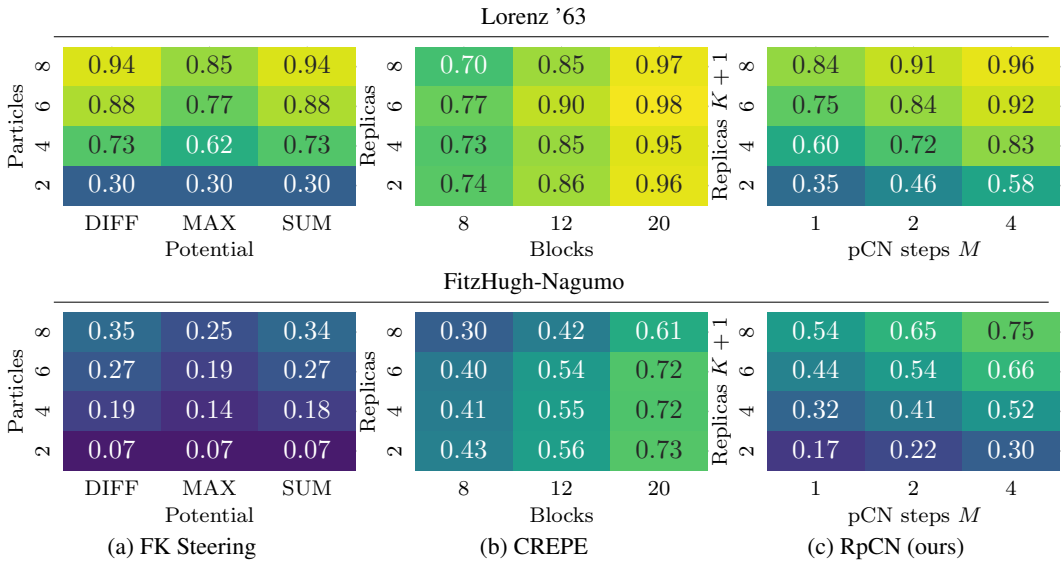
Diffusion Model Hyperparameters and Training We trained a diffusion model with the variance exploding probability path [42]. The diffusion model uses the UNet architecture from [14]. We use the flow matching loss [32] with the variance-exploding probability path [42], weighting the loss by $1/(\sigma'_{1-t})^2$ where σ_t is the variance-exlpoding noise standard deviation of $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$. The model is trained on 4,000 trajectories with batch size 500 for 2,000 epochs. We use the Adam optimizer with initial learning rate 1×10^{-4} and exponential decay learning rate scheduler with rate 0.995. For evaluation, we use an exponential moving average of the parameters computed during training with a smoothing factor of 1/400.

Method Hyperparameters While Finzi et al. [14] samples trajectories using 1,000 diffusion time steps, we only use $T+1 = 101$. FK Steering uses adaptive resampling [40], and the Euler-Maruyama sampler. The tilt for CREPE is annealed over u, where for a replica at sampling index u, the tilt strength is $\beta_u = (1-u)\beta$. CREPE does not use unadjusted Langevin correction steps ($M_{\text{lgvn}} = 0$), and does not use reward gradient guidance due to the reward not being differentiable. For RpCN, we choose replica tilts $\beta_k = \beta(k/K)^2$, and pCN step sizes $\lambda_k = 0.98$. Both CREPE and RpCN use the reverse diffusion sampler [42, Appendix E]. Fig. 7 shows an ablation of the other hyperparameters for each method. The ablation tests each method with the same range of memory usage (determined by the number of particles for FK Steering, and number of replicas for CREPE and RpCN). For this ablation, the maximum number of network function evaluations used by CREPE (attained when $M_{\text{blk}} = 20$) is slightly more than the maximum used by RpCN (attained when $K+1 = 8$ and $M = 4$. For Fig. 4, we selected the best hyperparameter set for each method.

Lorenz '63 The dynamical system introduced in [33] is given by

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix} = F(\boldsymbol{x}) = \begin{bmatrix}\sigma(x_2 - x_1)\\ x_1(\rho - x_3) - x_2\\ x_1x_2 - \beta x_3\end{bmatrix}$$

where chaotic behavior is induced when $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. Following [14], we instead use the scaled the system $f(\boldsymbol{x}) = F(20\boldsymbol{x})/20$ to ensure that $x_i(\tau) \in [-3, 3]$ for all dynamics time τ and $i \in [3]$. The event of a trajectory staying on one of the chaotic attractor is encoded by the constraint $C(\boldsymbol{x}) = 0.6 - \|\mathcal{F}[\boldsymbol{x} - \bar{\boldsymbol{x}}]\|_1 > 0$ where $\mathcal{F}$ is the Fourier transform over τ, $\|\cdot\|_1$ is the 1-norm over frequency magnitudes and system dimension, and $\bar{\boldsymbol{x}}$ is the mean of $\boldsymbol{x}(\tau)$ over τ.



(a) FK Steering (b) CREPE (c) RpCN (ours)

Figure 7: Success rate of FK Steering, CREPE, and RpCN varying hyperparameters.

FitzHugh-Nagumo The dynamical system introduced in [15, 36] models an excitable neuron and is given by

$$\dot{x}_i = x_i(a_i - x_i)(x_i - 1) - y_i + k\sum_{j=1}^{d} A_{ij}(x_j - x_i)$$

$$\dot{y}_i = b_i x_i - c_i y_i$$

for $i \in [2]$. The parameter values used are from [13, 14, 21]: $a_1 = a_2 = -0.025794$, $b_1 = 0.0065$, $b_2 = 0.0135$, $c_1 = c_2 = 0.2$, $k = 0.128$, and $A_{ij} = 1 - \delta_{ij}$ where δ_{ij} is the Kronecker delta. The neuron-spiking event is defined by the constraint $C(\boldsymbol{x}) = \max_\tau [x_1(\tau) + x_2(\tau)]/2 - 2.5 > 0$.

Computational Resources On one GPU, generating 32,000 samples for both FK Steering, CREPE, and RpCN takes less than one hour. This implies less than 36 hours of compute time in total.

B.3 Conditional Image Generation Experimental Details

Experimental Hyperparameters. Table 3 summarizes the shared inference and steering hyperparameters used for the conditional image generation experiments. Configurations for FK Steering are taken from [40].

Computational Resources To generate reward aligned samples for a single prompt with RpCN, it takes 393 seconds. For Table 1, we generate 100 samples, which split over 4 GPUS takes 163.8 minutes. Repeating this for the three models gives a total runtime of 491.3 minutes, around 8 hours.

B.4 Inverse Problem Solving Experimental Details

Noisy Inverse Problems Table 4 displays quantitative results for Bayesian (noisy) inverse problem ($\sigma = 0.05$) case study with Gaussian and Motion deblurring, comparing RpCN and DAS to DAPS and a variety of other generative inverse problem solvers. We find that although RpCN and DAS achieve strong performance in the noiseless setting, their performance deteriorates in the presence of measurement noise, particularly relative to DAPS. This is qualitatively shown in Figure 8 where DAPS generates a significantly sharper image than DAS and RpCN. This suggests that particle-based test-time methods tend to overfit noisy observations. Developing mechanisms to prevent this noisy overfitting is an important direction for future work.

Gradient Guidance For inverse problem solving, RpCN employs gradient guidance to better align the reverse transition kernel with the observed measurements. As described in FK Steering, rather

Category	Property	Value
Base model	Model Type	Stable Diffusion v1.4, v1.5, v2.1
Sampling (Shared)	Classifier-free Guidance Scale	7.5
	Batch Size	4
	Maximum Tilt	10
	Sampler (Default)	DDPM [8]
	Denoising Steps	100
	Resample Frequency	20
Sampling RpCN	pCN Steps	10
	Number of Replicas	4
Sampling CREPE	Number of Replicas	4
	Replica Placement (u)	$\{0, 1/3, 2/3, 1\}$
	Tilt annealing (β_u)	$\beta_u = (1-u)\beta$
	Sampling Steps Between Replicas	28
	Blocks (M_{blk})	3
Sampling DAS	Annealing $\min\{1, (1+\gamma)^t - 1\}$	$\gamma = 0.008$
	Resampling Schedule	Adaptive
	Resampler	SSP

Table 3: Shared inference and steering hyperparameters for conditional image generation (Table 1)

Operator	Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Deblur (Gaussian)	RpCN (Ours)	27.84	0.741	*0.211*	67.39
	DAS [29]	*27.88*	0.744	0.209	65.55
	DAPS [51]	**29.19**	**0.817**	**0.165**	**53.33**
	DPS [6]	25.87	0.764	0.219	79.75
	DDRM [28]	24.93	0.732	0.239	92.43
	DDNM [48]	28.20	0.804	0.216	57.83
	DCDP [31]	27.50	0.699	0.304	86.43
	FPS-SMC [10]	26.54	*0.773*	0.253	67.45
	DiffPIR [52]	27.36	–	0.236	*59.65*
Deblur (Motion)	RpCN (Ours)	*29.52*	0.769	*0.171*	49.55
	DAS [29]	29.58	0.773	0.169	47.24
	DAPS [51]	**29.66**	**0.847**	**0.157**	**39.49**
	DPS [6]	24.52	*0.801*	0.246	65.23
	DCDP [31]	25.08	0.512	0.364	125.13
	FPS-SMC [10]	27.39	0.826	0.227	*48.32*
	DiffPIR [52]	26.57	–	0.255	65.78

Table 4: Quantitative results for noisy inverse problem solving on FFHQ 256×256. Best results are in **bold**. Second best results are underlined. Third best results are *italicized*.



(a) Observation (b) DAPS (PSNR 28.58) (c) DAS (PSNR 27.03) (d) RpCN (PSNR 26.84)

Figure 8: Qualitative comparison of samples from RpCN, DAS and DAPS for noisy motion deblurring.

than sampling from the standard transition kernel

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_\theta(\boldsymbol{x}_{t+1}, t+1), \tau_{t+1}^2 \boldsymbol{I})$$

we twist the kernel by incorporating the gradient of the reward function, yielding the guided transition

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_\theta(\boldsymbol{x}_{t+1}, t+1) + g(t)\nabla_{\boldsymbol{x}_t} R(\boldsymbol{x}_t), \tau_{t+1}^2 \boldsymbol{I})$$

where $g(t)$ is a time-dependent schedule encoding the strength of guidance.

$$g(t) = \begin{cases} 0.0 & t \geq 500 \\ 1.0 & t < 500 \end{cases}$$

This time-dependent schedule aligns the data with the gradient during the later stages of denoising, avoiding the instabilities when the signal is unreliable during initial denoising steps. We refer to this gradient strength schedule as *late guidance*.

To promote additional numerical stability, the gradient is ℓ_2-normalized prior to being applied. This normalization is applied to both RpCN and DAS for maximum comparability.

Experimental and Operator Hyperparameters Table 5 summarizes inference and steering hyperparameters used by RpCN for inverse problem solving. Configurations DAPS and for all other methods can be found in [51]. Additionally, to enable a fair comparison to DAPS, we match their observation operator hyperparameters for noisy and noiseless inverse problem solving. Details are summarized in Table 6.

Category	Property	Value
Base model	DDPM	FFHQ (Pretrained)
Sampling (Shared)	Denoising Steps	1000
	Batch Size	1
	Gradient Normalization	Yes
RpCN Details	Replicas	3
	PCN Steps	10
	Resample Frequency	25
	Maximum Tilt	1
	Gradient Guidance Schedule	Late
DAS Details	Replicas	3
	Annealing $\min\{1, (1+\gamma)^t - 1\}$	$\gamma = 0.008$
	Resampling Schedule	Adaptive
	Resampler	SSP

Table 5: Inference and steering configuration for RpCN for inverse problem solving.

Operator	Configs
Deblur (Gauss)	Kernel size 61, Intensity 3.0
Deblur (Motion)	Kernel size 61, Intensity 0.5
Super Resolution $4\times$	Scale factor 4, Resolution 256
Random Inpainting	Mask probability range $[0.70, 0.71]$, Resolution 256

Table 6: Inference and steering configuration for RpCN for noiseless inverse problem solving.

Computational Resources To generate reward aligned samples for a single prompt, it takes 442.0 seconds, about 7 minutes. For Table 2, we generate 100 samples, which split over 4 GPUS takes 184.2 minutes. Repeating this for the 4 operators gives a total runtime of 736.8 minutes, around 12 hours.