ACCELERATING DISCRETE DIFFUSION MODELS WITH PARALLEL SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete diffusion models are widely used for learning and generating discrete distributions. As the generation process is inherently sequential, the acceleration of sampling is of significant importance. In this work, we parallelize the mainstream τ-leaping algorithm for absorbing discrete diffusion in a Continuous-Time Markov Chain (CTMC) framework. By leveraging the continuous-time stochastic integral form of the τ-leaping algorithm and the Picard iteration method, we achieve parallel-in-time sampling acceleration. We implement a predictor-corrector structure based on the Markov chain Monte Carlo (MCMC) method to control for additional errors and provide a proof of exponential convergence for our algorithm. We improve the overall time complexity of τ-leaping from $\mathcal{O}(d^2 \log^2 d)$ to at most $\mathcal{O}(d^{3/2} \log^{5/2} d)$. In practice, our accelerated algorithm can achieve at most 5 to 8-fold sampling speedup over the traditional time-sequential τ-leaping with respect to wall-clock time on convex, non-convex, and high-dimensional discrete distributions. Our research broadens the scope of leveraging discrete diffusion models in various challenging areas like molecular structure generation and Large Language Models (LLMs).

1 INTRODUCTION

Diffusion models over discrete spaces (Austin et al., 2021) have become central to generative modeling for categorical data, with increasing impact across applications including molecular design (Vignac et al., 2022), protein engineering (Gruver et al., 2023), DNA sequence generation under biophysical constraints (Sarkar et al., 2024), and high-fidelity generation of text (Zheng et al., 2023), music (Yang et al., 2023), and images (Lezama et al., 2022).

However, due to the sequential nature of the sampling process, to generate a high-quality sample via diffusion models often requires a large number of sequential rounds, and sequential evaluation of the neural network-based score function with expensive time cost (Song et al., 2020). However, the algorithms underlying the above results are highly sequential and fail to fully exploit contemporary parallel computing resources such as multi-core central processing units (CPUs) and many-core graphics processing units (GPUs).

The primary focus of existing parallel sampling algorithms is on parallel-in-tokens (Wu et al., 2025; Xie et al., 2025). While parallel-in-time algorithms for continuous diffusion models have been extensively studied (Shih et al., 2023; Anari et al., 2024; Chen et al., 2024a; Zhou & Sugiyama, 2025), their counterparts for discrete diffusion models remain largely unexplored.. This gap motivates our investigation into the question:

Whether parallelization can fundamentally accelerate sampling for discrete diffusion models?

In this paper, we make significant progress in both theoretical and practical sides. The **key contributions** of this work are summarized as follows:

- **A fast parallel sampler.** We introduce the first *parallel-in-time* τ-leaping algorithm (Cao et al., 2006) for discrete diffusion inference, namely the Picard τ-leaping method. Our method has been shown as a fast sampler that reduces the time complexity from $\widetilde{\mathcal{O}}(d^2)$

Table 1: Comparisons of our parallel methods and sequential methods for τ-leaping given δ^2-accuracy of the score. Here, S denotes the size of vocabulary.

	Time Complexity	Space Complexity
(Campbell et al., 2022, Theorem 1)	$\mathcal{O}(\frac{d^4}{\delta})$	$\mathcal{O}(d)$
(Ren et al., 2025, Theorem 4.7)	$\mathcal{O}(\frac{d^2 \log^2(d/\delta^2)}{\delta^2})$	$\mathcal{O}(d)$
(Liang et al., 2025, Theorem 1)	$\mathcal{O}(\frac{d^2 S}{\delta^2})$	$\mathcal{O}(d)$
Ours, Theorem 3.1	$\mathcal{O}(\frac{d^{3/2} \log^{5/2}(d\delta^{-2})}{\delta})$	$\mathcal{O}(d^{3/2} \log^{1/2}(d))$

to $\widetilde{\mathcal{O}}(d^{3/2})$ with slightly larger space complexity[1] (See Theorem 3.1). Here, $\widetilde{\mathcal{O}}$ omits the logarithmic factors. We summarize the comparison between existing sequential methods and our results in Table 1.

- **Practical parallel sampling via Corrector.** We also propose two Markov Chain Monte Carlo (MCMC)-based correctors designed to address sampling in both simple and complex scenarios for further mitigating the parallel iteration and discretization error with detailed convergence analysis (See Section 4).
- **Experimental evaluation.** Our method achieves substantial improvements in sampling efficiency over the original sequential algorithm, measured by both the Number of Function Evaluations (NFE) and wall-clock runtime in experiments (See Section 5).

In Appendix B, we discuss more related works.

2 Preliminaries on Discrete Diffusion Models

In discrete diffusion models, the forward process is a continuous-time Markov chain (CTMC) $(\boldsymbol{x}_t)_{t\in[0,T]}$ on a finite state space $\mathbb{X}$. Let $\boldsymbol{p}_t \in \Delta^{|\mathbb{X}|}$ denote the law of $\boldsymbol{x}_t$ as a column vector. The forward equation is

$$\frac{\mathrm{d}\boldsymbol{p}_t}{\mathrm{d}t} = \boldsymbol{Q}_t \boldsymbol{p}_t, \qquad \boldsymbol{Q}_t = \big(\boldsymbol{Q}_t(y,x)\big)_{x,y\in\mathbb{X}}, \tag{1}$$

where $\boldsymbol{Q}_t$ is a rate matrix with (i) $\boldsymbol{Q}_t(x,y) \geq 0$ for $x \neq y$ and (ii) $\boldsymbol{Q}_t(x,x) = -\sum_{y\neq x} \boldsymbol{Q}_t(y,x)$. We write $\widetilde{\boldsymbol{Q}}_t := \boldsymbol{Q}_t - \mathrm{diag}(\boldsymbol{Q}_t)$ for the off-diagonal part. The time-reversed (backward) process $\big(\overleftarrow{\boldsymbol{x}}_t\big)_{t\in[0,T]}$, with $\overleftarrow{*}_t := *_{T-t}$, is again a CTMC with law $\overleftarrow{\boldsymbol{p}}_s$ and generator $\overleftarrow{\boldsymbol{Q}}_t$ satisfying (Kelly, 2011)

$$\frac{\mathrm{d}\,\overleftarrow{\boldsymbol{p}}_t}{\mathrm{d}t} = \overleftarrow{\boldsymbol{Q}}_t \overleftarrow{\boldsymbol{p}}_t, \tag{2}$$

where for $x \neq y$, $\overleftarrow{\boldsymbol{Q}}_t(y,x) = \frac{\overleftarrow{\boldsymbol{p}}_t(y)}{\overleftarrow{\boldsymbol{p}}_t(x)} \overleftarrow{\boldsymbol{Q}}_t(x,y)$, and $\overleftarrow{\boldsymbol{Q}}_t(x,x) = -\sum_{y'\neq x} \overleftarrow{\boldsymbol{Q}}_t(y',x)$, and $\overleftarrow{\boldsymbol{Q}}_t := \boldsymbol{Q}_{T-t}$.

According to Proposition 3.2 in (Ren et al., 2025), discrete diffusion models can also be interpreted as stochastic integrals with Poisson random measure. The forward process in discrete diffusion models (1) can thus be represented by the following stochastic integral:

$$\boldsymbol{x}_t = \boldsymbol{x}_0 + \int_0^t \int_{\mathbb{D}} \nu N[\lambda](\mathrm{d}t, \mathrm{d}\nu), \tag{3}$$

where the intensity λ is defined as $\lambda_t(\nu,\omega) = \overleftarrow{\boldsymbol{Q}}_t(x_{t^-}(\omega)+\nu, x_{t^-}(\omega))$ if $x_{t^-}(\omega)+\nu \in \mathbb{X}$ and 0 otherwise. Here, the outcome $\omega \in \Omega$ and x_{t^-} denotes the left limit of the càdlàg process x_t at time t with $x_{0^-} = x_0$. We will also omit the variable ω, should it be clear from context.

The backward process in discrete diffusion models (2) can also be represented similarly as

$$\boldsymbol{y}_t = \boldsymbol{y}_0 + \int_0^t \int_{\mathbb{D}} \nu N[\mu](\mathrm{d}s, \mathrm{d}\nu), \tag{4}$$

[1]We note, in this paper, that the space complexity refers to the number of words (Cohen-Addad et al., 2023) instead of the number of bits (Goldreich, 2008) to denote the approximate required storage.

where the intensity μ is defined as

$$\mu_t(\nu, \omega) = \overleftarrow{s}_t(\boldsymbol{y}_{t^-}, \boldsymbol{y}_{t^-} + \nu)\overleftarrow{\boldsymbol{Q}}_s(\boldsymbol{y}_{s^-}, \boldsymbol{y}_{s^-} + \nu) \tag{5}$$

if $y_{s^-} + \nu \in \mathbb{X}$ and 0 otherwise. During inference,

$$\widehat{\boldsymbol{y}}_t = \widehat{\boldsymbol{y}}_0 + \int_0^t \int_{\mathbb{D}} \nu N[\widehat{\mu}](\mathrm{d}s, \mathrm{d}\nu) \tag{6}$$

is used instead of Equation (4), where the estimated intensity $\widehat{\mu}$ is defined by replacing the true score $\boldsymbol{s}_t$ with the neural network (NN) estimated score $\widehat{\boldsymbol{s}}_t$ in Equation (5).

3 Parallel Sampling for Absorbing Discrete Diffusion Models

In this section, we introduce the parallel-in-time τ-leaping algorithm for discrete diffusion models in a CTMC framework. The key idea is to extend the application of Picard iteration (See Appendix C.2) from the domain of diffusion models in continuous state spaces to the stochastic integral form of τ-leaping (Cao et al., 2006) in discrete state spaces.

This section is organized as follows. Section 3.1 introduces and details our parallel algorithm based on Picard iteration. Section 3.2 then states the assumptions and their justifications that are necessary for the subsequent theoretical analysis. Finally, Section 3.3 presents the main theorem regarding the algorithm's error bound and computational complexity, followed by its proof.

3.1 Our Algorithm

Discretization Scheme. In this work, the total time horizon T is firstly segmented in to N blocks with the large time grid $(t_n)_{n\in[0,N]}, t_0 = 0, t_N = T$. For each block, the time interval $[t_n, t_{n+1}]$ is divided into M steps with the small time grid $(\tau_{n,m})_{n\in[0,N-1],m\in[0,M]}, \tau_{n,0} = t_n$, and $\tau_{n,M} = t_{n+1}$. We also define the small step size $\epsilon = \tau_{n,m} - \tau_{n,m-1}$. It is also possible to set $M = M_n$ and $\epsilon = \epsilon_n$ for flexible grid when using early stopping.

When describing the standard τ-leaping algorithm with the form in (equation 6), the main update step in Picard iteration can be described as

$$\widehat{y}^{(k+1)}_{t_n+m\epsilon} = \widehat{y}_{t_n} + \sum_{j=0}^{m-1} \left(\sum_{y'\in\mathbb{X}} (y' - \widehat{y}^{(k)}_{t_n+j\epsilon}) \cdot \mathcal{P}(\widehat{\mu}^{\theta}_{t_n+j\epsilon}(y'|\widehat{y}^{(k)}_{t_n+j\epsilon}) \cdot \epsilon) \right), \tag{7}$$

where $\widehat{y}^{(k)}_{t_n+j\epsilon}$ is the sample state a $t = t_n + j\epsilon$ in the k-th Picard iteration. $\mathcal{P}(\widehat{\mu}^{\theta}_{t_n+j\epsilon}(y'|\widehat{y}^{(k)}_{t_n+j\epsilon})$ is the numerical form of $N[\widehat{\mu}^{\theta}_{\lfloor\cdot\rfloor}]$, which denotes the number of jumps from $\widehat{y}^{(k)}_{t_n+j\epsilon}$ to y' during the small time interval ϵ. In the block-wise parallel algorithm, the computation for each block starts from its initial state $\widehat{y}_{t_n}$ which is also the terminal value of the last block.

One may notice that such update includes a summation among all possible adjacent states, i.e. $\sum_{y'\in\mathbb{X}}$, which would lead to an exponential computation complexity with respect to the data dimension d. In practice, the computation cost is controlled at $\mathcal{O}(Sd)$ with the vocabulary size S, using techniques such as dimension factorization (Campbell et al., 2022) and Top-K candidate set truncation (Ye et al., 2024). Our analysis is also based on such settings.

The Picard method transforms the dependence on the previous time step in the τ-leaping process into a dependence on the entire trajectory of the previous iteration. The state of the new trajectory at time point $t_n + m\epsilon$ is equal to the initial state of the block $\widehat{y}_{t_n}$, plus the sum of all jump vectors from $j = 0$ to $j = m - 1$ in the last iteration which is fully known, therefore the calculation of jump for all time steps within a block can be fully parallelized with cumulative sum operation using *parallel prefix sum* algorithm, eliminating the need for a sequential loop. The overall computation process is summarized in Algorithm 1.

One may notice that there is an optional corrector added after computing one block. Such a module can compensate for the additional error introduced by the Picard iteration for a minor increase in the number of function evaluations (NFE). This, in turn, relaxes the constraints on the parameters of the parallel algorithm, such as the number of blocks N or small intervals M. A more detailed analysis will be presented in Section 4. It is important to note that the following analysis in Section 3.3 will focus exclusively on the parallel τ-leaping method itself without considering the corrector.

Algorithm 1: Parallel τ-Leaping Algorithm for Discrete Diffusion Model Sampling

Input : $\widehat{y}_0 \sim q_0$, large time grid $(t_n)_{n\in[0,N]}$ with $t_0 = 0$, $t_N = T - \xi$, small time grid $(\tau_{n,m})_{n\in[0,N-1],m\in[0,M]}$ with $\tau_{n,0} = t_n$, $\tau_{n,M} = t_{n+1}$ and small step size $\epsilon = \tau_{n,m+1} - \tau_{n,m}$; Picard depth K_p; intensity $\widehat{\mu}_s^\theta$; score estimate $\widehat{s}_t^\theta$; fixed random seeds.

Output: A sample $y_{t_N} \sim q_{t_N}$.

for $n = 0$ **to** $n = N - 1$ **do**

  Initial Guess: $\widehat{y}^{(0)}_{t_n+m\epsilon} \leftarrow \widehat{y}_{t_n}$ for all $m \in [0, M]$

  for $k = 0$ **to** $K_p - 1$ **do**

    for $j = 0$ **to** $M - 1$ *Parallel* **do**

      $\widehat{\mu}_j^\theta \leftarrow \widehat{\mu}^\theta_{t_n+j\epsilon}\big(\cdot\,|\widehat{y}^{(k)}_{t_n+j\epsilon}\big)$

      $J_{j,y'} \sim \mathcal{P}(\widehat{\mu}_j^\theta(y')\cdot\epsilon)$ for all $y' \in \mathcal{X}_{neighbor}$

      $\Delta\widehat{y}_j^{(k)} = \sum_{y'\in\mathbb{X}_{neighbor}} \big(y' - \widehat{y}^{(k)}_{t_n+j\epsilon}\big)\cdot J_{j,y'}$

    end

    for $m = 1$ **to** M *Parallel* **do**

      $\widehat{y}^{(k+1)}_{t_n+m\epsilon} \leftarrow \widehat{y}_{t_n} + \sum_{j=0}^{m-1} \Delta\widehat{y}_j^{(k)}$

    end

  end

  $y_{t_{n+1}} \leftarrow \widehat{y}^{(K_p)}_{t_n+M\epsilon}$

  $\widehat{y}_{n+1} \leftarrow \mathsf{Corrector}(y_{t_{n+1}}, \widehat{s}_t^\theta)$ (Optional)

end

3.2 Assumption

We first list the assumptions that will be used in the subsequent analysis and proofs, which mainly follows the settings in (Ren et al., 2025) due to the shared CTMC and stochastic integral frameworks. These assumptions align with those established in previous theoretical works, such as those described by (Ren et al., 2025).

Assumption 3.1 (Bounded Score and Rate Matrix). The true score function $s_t(x, y)$, learned score function $\widehat{s}_t^\theta(x, y)$, and the symmetric time-homogeneous rate matrix Q satisfy:
(i) $\widehat{s}_t^\theta(x, y) \in [m_s, M_s]$ and $s_t(x, y) \lesssim 1 \vee t^{-\gamma}$, for $\forall x, y \in \mathbb{X}$, $t > 0$, and a constant $\gamma \in [0, 1)$.
(ii) $Q(x, y) \le C$ and $|Q(x, x)| = |-\sum_{y\neq x} Q(x, y)| \le \overline{D}$ for $\forall x, y \in \mathbb{X}$, $t > 0$ and some constants $C, \overline{D} > 0$.
(iii) $\rho(Q)$ as the modified log-Sobolev constant of Q has a lower,bound $\rho > 0$.

Assumption 3.2 (Score Estimation Accuracy). The neural network $\widehat{s}_t^\theta(x_t)$ learned the true score $s_t(x_t, t)$ with δ-accuracy, which is described as

$$\sum\nolimits_{n=0}^{N-1}(t_{n+1} - t_n)\mathbb{E}\bigg[\int_{\mathbb{X}} K\bigg(\frac{\widehat{s}^\theta_{t_n}(x_{t_n}, y)}{\widehat{s}_{t_n}(x_{t_n}, y)}\bigg)\widehat{s}_{t_n}(x_{t_n}, y)\widetilde{Q}(x_{t_n}, y)\nu(\mathrm{d}y)\bigg] \le \delta^2,$$

where $K = x - 1 - \log x$ and $\widetilde{Q}$ denotes the matrix Q with the diagonal elements set to zero.

Assumption 3.3 (Continuity of Score Function). For any $t > 0$ and $y \in \mathbb{X}$ such that $Q(x_{t^-}, y) > 0$, we have

$$\bigg|\frac{\mu_{t^+}(y)}{\mu_t(y)}\bigg| := \bigg|\frac{p_t(x_{t^-})Q(x_t, y)}{p_t(x_t)Q(x_{t^-}, y)} - 1\bigg| \lesssim 1 \vee t^{-\gamma},$$

for some exponent $\gamma \in [0, 1)$ and t^+, t^- denotes the right and left limits.

Remark 3.1. For simplicity, we do not consider the early-stopping in our framework, which requires $\gamma < 1$ according to Proposition C.5 in (Ren et al., 2025).

Assumption 3.4 (Lipschitz Continuity). Given the learned intensity function $\widehat{\mu}_t^\theta(y|x) = \widehat{s}_t^\theta(x, y)\widetilde{Q}(x, y)$ and the small step size ϵ, for $\forall x_1, x_2 \in \mathbb{X}$, the following Lipschitz condition with respect to the Poisson expectation holds:

$$\sum_{y'\in\mathbb{X}} \mathbb{E}[(\mathcal{P}(\widehat{\mu}_t^\theta(y'|x_1)\cdot\epsilon) - \mathcal{P}(\widehat{\mu}_t^\theta(y'|x_2)\cdot\epsilon))^2] \le (L_\mu\epsilon)^2\|x_1 - x_2\|^2.$$

Assumption 3.4 is newly introduced which plays an important role in deriving the convergence of Picard iteration as shown in (Anari et al., 2024; Chen et al., 2024a; Zhou & Sugiyama, 2025). Since the intensity function is constructed by the learned score and the non-diagonal rate of the forward process (Ren et al., 2025), and both two components have continuity or bounded properties based on Section 2 and Assumption 3.3, the setting could be reasonable.

3.3 Theoretical Guarantees

Next, we provide an approximated error bound of our algorithm without correctors.

Theorem 3.1 (Theoretical Guarantees for Parallel τ-Leaping). *Under the assumptions above in this section, the Picard τ-leaping algorithm (Algorithm 1) achieves the following approximation error bound:*

$$D_{KL}(p_0 \| \widehat{q}_T^{(K_p)}) \lesssim \underbrace{e^{-\rho T} \log |\mathbb{X}|}_{\text{truncation error}} + \underbrace{\delta^2}_{\text{score estimation error}} + \underbrace{\overline{D}^2 \epsilon T}_{\text{discretized error}} + \underbrace{C^{K_p} \text{poly}(\overline{D})}_{\text{parallization error}}$$

with constant $C < 1$. Furthermore, by choosing parameters of the following order and with assuming $\overline{D} = \mathcal{O}(d)$ (Ren et al., 2025),

$$T = \mathcal{O}(\log(d\delta^{-2})), \quad \epsilon = \mathcal{O}(\delta^2 d^{-2} \log^{-1}(d\delta^{-2})), \quad K_p = \mathcal{O}(\log(d\delta^{-2})),$$
$$h = \mathcal{O}(\delta d^{-3/2} \log^{-1/2}(d\delta^{-2})), \quad N = T/h = \mathcal{O}(\delta^{-1}(d^{3/2} \log^{5/2}(d\delta^{-2}))),$$

the algorithm achieves $NK = \mathcal{O}(\delta^{-1} d^{3/2} \log^{5/2}(d\delta^{-2})) = \widetilde{\mathcal{O}}(\delta^{-1} d^{3/2})$ approximate time complexity and $dM = dh/\epsilon = \mathcal{O}(\delta^{-1} d^{3/2} \log^{1/2}(d\delta^{-2}))$ space complexity.

The meaning of notations is shown in Algorithm 1. Compared with the sequential τ-leaping, our parallel algorithm improves the approximate time complexity from $\widetilde{\mathcal{O}}(d^2)$ to $\widetilde{\mathcal{O}}(d^{3/2})$ with the cost of increasing $\mathcal{O}(d^{1/2} \log^{1/2}(d))$ space complexity. The proof can be found in Appendix D.

The main obstacle to improving the parallel complexity to $\text{poly}\log(d)$ is that even under Lipschitz condition (Assumption 3.4), ensuring convergence of the Picard iteration requires the length of time slice scales as $\delta/(d^{3/2} L_p)$ rather than $1/L_p$ in continuous case where L_p denotes the smoothness of the intensity.

Remark 3.2 (Practical choices of the depth). Following the practical settings of Picard-based parallel algorithms in continuous diffusion models (Shih et al., 2023), one may choose depth from 2 to 12 depending on the complexity of the problem and See Section 5.1.1 for experimental results.

4 Parallel Sampling with Correctors

Although Theorem 3.1 provides an error bound for the parallel sampling algorithm, the block width h cannot achieve $O(1)$ with respect to d. Consequently, in practical applications, the selection of the width or the number of parallel steps is more restricted compared to parallel continuous diffusion (Chen et al., 2024a). To address this limitation, in this section, we propose two MCMC-based Plug-and-Play correctors to refine the generated distribution after the parallel algorithm completes the computation of a block. We provide a general error analysis of the corrector described in the following theorem.

Theorem 4.1 (Error Control of the MCMC Corrector). *Suppose $\bar{p}_{t_n}$ is the true target distribution derived from the forward process, and $\widehat{q}_{t_n}$ is the generated distribution from our predictor-corrector algorithm 2 and 3. The MCMC-based corrector with stationary distribution $\bar{p}_{t_{n+1}}$ described has the following rate of error control:*

$$E_{n+1} := TV(\widehat{q}_{t_{n+1}}, \bar{p}_{t_{n+1}}) \leq \gamma^{K_c}(E_n + \Delta_p) + \Delta_c, \tag{8}$$

where $\bar{p}_{t_n}$ is the the true distribution and $\widehat{q}_{t_n}$ is the generated distribution at time t_n. Δ_p denotes the discretization error from the predictor, Δ_c is the corrector approximation error from learned score, $\gamma < 1$ is the contraction coefficient related to the conductance of the problem, and K_c is the number of corrector steps.

The proof is demonstrated in Appendix E.1.

Remark 4.1 (**Comparison with Theorem 3.1**). Since the theoretical guarantees analyzed in Theorem 3.1 do not take the corrector into consideration, the error analysis in Theorem 4.1 is independent to the analysis in Theorem 3.1. However, based on the experimental results in Section 5, we believe that the condition and bound in Theorem 3.1 could be relaxed and improved after adding correctors to the proof framework,which will be explored in future work.

4.1 METROPOLIS–HASTINGS CORRECTOR

We first introduce a Metropolis–Hastings Algorithm-based MCMC corrector with symmetric proposal distribution for experiments in the next section. The computation framework is summarized in Algorithm 2.

Algorithm 2: Metropolis–Hastings Algorithm as Corrector ($\text{Corrector}_{\text{MH}}(y_{t_{n+1}}, \widehat{s}_t^\theta)$)

Input : Sample from the predictor $y_{\text{pred}} = y_{t_{n+1}}$,
symmetric proposal distribution $Q(y'|y) = Q(y|y')$, score estimate $\widehat{s}_t^\theta$.
Output: Corrected sample $\widehat{y}_{t_{n+1}}$
Initialization: $y_{\text{current}} \leftarrow y_{\text{pred}}$
for $k = 0$ **to** K_c **do**
  $y_{\text{proposed}} \sim Q(y'|y_{\text{current}})$
  $A \leftarrow \min\left(1, \widehat{s}^\theta_{t_{n+1}}(y_{\text{current}}, y_{\text{proposed}})\right)$
  $u \sim \text{Unif}([0,1])$
  if $u < A$ **then**
    $y_{\text{current}} \leftarrow y_{\text{proposed}}$
  end
end

Given the output of the predictor after one block sampling $y_{t_{n+1}} = \widehat{y}^{(K_p)}_{t_n + M\epsilon} := y_{pred}$, the corrector runs a short, serial Metropolis-Hastings (MH) MCMC process using $\bar{p}_{t_{n+1}}$, the true distribution at t_{n+1}, as the stationary distribution. The MH corrector accepts or rejects new proposed states according to the probability distribution at the current timestep t_{n+1} provided by the learned score function. This process pulls $y_{t_{n+1}}$ from potentially low-probability regions back into high-probability ones, effectively correcting the error.

Lemma 4.1 (**Acceptance rate with symmetric proposal distributions**). Given the estimated score $\widehat{s}_t^\theta$ with the symmetric proposal distribution $Q(y'|y) = Q(y|y')$, the acceptance rate of changing from y to y' the Metropolis-Hastings MCMC corrector can be approximated as

$$A(y'|y) \approx \min(1, \widehat{s}_t^\theta(y, y')), \tag{9}$$

The lemma above can be easily proved according to the definition of acceptance rate and symmetric proposal distributions Q:

$$A(y'|y) = \min\left(1, \frac{\pi(y')Q(y|y')}{\pi(y_i)Q(y'|y)}\right) = \min\left(1, \frac{\pi(y')}{\pi(y_i)}\right).$$

where $\pi(\cdot) = \bar{p}_{t_{n+1}}$ is our target stationary distribution. Then, according to the definition of concrete score in (Lou et al., 2024):

$$\frac{\bar{p}_{t_{n+1}}(y')}{\bar{p}_{t_{n+1}}(y)} = \bar{s}_{t_{n+1}}(y, y'),$$

we can naturally use the learned score as the approximated acceptance rate.

The setting of symmetric proposal works well in low-dimension cases (see Section 5). However it makes the corrector difficult to work in more complex situations such as language models because such symmetry does not naturally hold, and the proposal sample requires a uniform sampling in the entire vocabulary which leads to slow mixture and acceptance rate (Levin & Peres, 2017).

4.2 TOP-K BARKER CORRECTOR

To adapt the corrector to more complex cases, with the example of absorbing diffusion, we introduce an MCMC corrector with Uniform Top-K Proposal and Barker Acceptance (Livingstone & Zanella, 2022) in Algorithm 3.

Algorithm 3: Top-K Barker Algorithm as Corrector ($\mathsf{Corrector}_{\mathsf{Barker}}(y_{t_{n+1}}, \widehat{s}_t^\theta)$)

Input : Current time t, vocabulary $\mathcal{V} = \{1, \dots, n\}$,$x_{\text{pred}} = (x_1, \dots, x_d) \in \mathcal{V}^d$, score estimate $\widehat{s}_t^\theta$, candidate count K_b, subset of positions to correct $\mathcal{I} \subseteq \{1, \dots, d\}$, absorbing token $M' \in \mathcal{V}$

Output: Corrected sample $\widehat{y}_{t_{n+1}}$

Create masked copy: $\widetilde{x} = x^{(i \leftarrow M')}$ such that $\widetilde{x}_i \leftarrow M'$ for $\forall \in \mathcal{I}$

$r_{i,j} \leftarrow \widehat{s}_t^\theta(x^{(i \leftarrow M)}, x^{(i \leftarrow j)})$ for $\forall i \in \mathcal{I}, j \in \mathcal{V}$

Compute unnormalized log-weights: $u_{i,j} \leftarrow \log r_{i,j}$

for $i \in \mathcal{I}$ **do**

  Sort $j \in \mathcal{V}$ by $u_{i,j}$ descending and select top K_b indices as $\mathcal{K}_i$

  if $a := x_i \notin \mathcal{K}_i$ **then**

    Replace the last element of $\mathcal{K}_i$ by a

  end

  $b \sim q_i(\cdot)$, where $q_i(j) = 1/K_b, j \in \mathcal{K}_i$

  $\Delta_i \leftarrow u_{i,b} - u_{i,a}$

  $\alpha_i \leftarrow \frac{1}{1+\exp(-\Delta_i)}$

  $u \sim \text{Unif}[0, 1]$

  if $U \leq \alpha_i$ **then**

    $x_i \leftarrow b$

  end

end

The method uses the network's outputs to score single-site replacements, but builds proposals in a MASK-anchored context so that the proposal is independent of the current token.

A Top-K truncation focuses computation on promising candidates, and a uniform or softmax proposal over this set keeps the step cheap while preserving correctness. We use the Barker acceptance, yielding a smooth decision that satisfies the detailed balance condition. So the corrector provably leaves the reverse-time marginal invariant rather than introducing a bias. The entire correction round requires only one network forward for a batch of positions and can be fully parallelized.

Definition 4.1 (Single-Coordinate Barker Kernel). Let $x = \{x_1, \dots, x_d\} \in \mathbb{X}$ be the state, and $\mathbb{X} = \mathcal{V}^d$ be a finite state space with vocabulary $\mathcal{V} = \{1, \dots, n\}$ and data dimension d. For absorbing diffusion, we set the mask token as M'. For a position $i \in \{1, \dots, d\}$ and element (token) $j \in \mathcal{V}$, we denote $x^{(i \leftarrow j)}$ as the state after replacing x_i with j. Suppose the target distribution is $\pi_t : \mathbb{X} \to (0, 1]$ with full support. Define $u_{i,j}(x) = \widehat{s}_t^\theta(x, x^{(i \leftarrow j)})$ given a trained score $\widehat{s}_t^\theta$, and an anchor context $\widetilde{x}^{(i)} := x^{(i \leftarrow M')}$. Construct a candidate set $\mathcal{K}_i(\widetilde{x}^{(i)}) \subseteq \mathcal{V}$ of fixed size K_b, and define a proposal distribution $q_i(\cdot|x)$ supported and uniform on it. We define a single-coordinate Barker kernel $K_i(x, x')$. Given x, one first sample $b \sim q_i(\cdot|x)$ and construct $x' = x^{(i \leftarrow b)}$. Then accept x' with probability

$$\alpha_i(x \to x') = \frac{1}{\exp(-\Delta_i)}, \quad \Delta_i = u_{i,b} - u_{i,a},$$

where $a := x_i$ is the current token.

The designed kernel has the property described in Theorem 4.2.

Theorem 4.2 (Invariance of the Top-K Barker Corrector Kernel). *For all $i \in \{1, \dots, d\}$, the Markov kernel K_i is π_t-reversible and has π_t as the stationary distribution satisfying the detailed balance condition.*

$$\pi_t(x)K_i(x, x') = \pi_t(x')K_i(x', x). \tag{10}$$

See Appendix E.2 for the proof. With the proposition above, Algorithm 3 can then be incorporated into the error analysis framework of Theorem 4.1.

5 EXPERIMENTS

In this section, we evaluate the performance of our parallel τ-leaping algorithm against the sequential τ-leaping algorithm. The evaluation is conducted on four synthetic distributions shown in Table 2, image generation based on Imagenet, and text generation based on OpenWebText.

5.1 SYNTHETIC EXPERIMENTS

Oracle Models To directly verify the speedup of our parallel τ-leaping algorithm over its sequential counterpart, we employ an Oracle Model. This model provides the sampler with perfect, analytically tractable score information, thereby eliminating approximation errors from the score network. Consequently, the quality of the final samples serves as a direct measure of the sampling algorithm's intrinsic precision and stability.

Controlled Variables The essence of the Parallel-in-time acceleration strategy is to partition the original total number of serial time steps U into N blocks. The U/N time steps within each block are then computed through parallel iterations, thereby reducing the total amount of serial computation and the Number of Function Evaluations (NFE).

For a fair comparison, we fix the number of correction steps K_c, parallel blocks N, and steps per block M in each experiment. The sequential baseline algorithm runs for a total of $M \times N$ steps, with a correction applied every M steps. The number of samples is controlled to avoid GPU memory bottlenecks, and all experiments utilize the Metropolis-Hastings corrector.

We conduct experiments on sampling 4 types of synthetic distributions: Chessboard, Circle, Hypercube, and Embedded Hypercube. The feature and settings of parameters are demonstrated in Table 2. One can check more detailed explanation in Appendix F.

Table 2: Characteristics and Parameter Settings of Experimental Distributions. Runtimes and KL Divergence for all experiments are averaged over 20 runs.

Distribution	**Characteristics**	**Dimensions/Grid**	**N**	**M**	K_c	**Samples**
Chessboard	Sparse, Multi-modal	8×8	40	50	5	8196
Circle	Non-Convex, Connected	32×32	60	50	8	4096
Hypercube	High-dimension	6-d	100	50	10	2048
Embed-Hypercube	Low-dimension manifold	6-d in 12-d	50	100	8	2048

5.1.1 PERFORMANCE

Given that the Picard iteration depth K_p is a parameter unique to the parallel sampling algorithm, we evaluate its performance at various K_p values against the sequential algorithm. The comparison is based on three key metrics: Wall-Clock Runtime, NFE, and the KL Divergence of the generated distribution. Table 3 shows the comprehensive performance comparison across four baseline distributions.

As can be seen from the table, our parallel predictor-corrector algorithm significantly reduces the Number of Function Evaluations (NFE) and clock-wall runtime while maintaining high accuracy across the vast majority of experimental settings. However, we also observe a diminishing marginal return with a continued increase in the iteration depth K_p. Furthermore, when the batch size is large, increasing the number of parallel iteration steps leads to higher data I/O and communication cost. Once the GPU bandwidth bottleneck is reached, the efficiency of the parallel algorithm will decrease. Therefore, in practical applications, a moderate iteration depth is sufficient.

Table 3: Performance comparison on four distributions. Top: Performance on Chessboard and Circle distributions. Bottom: Performance on Hypercube and Embedded Hypercube distributions.

Alg.	K_p	**Chessboard**			**Circle**		
		Runtime (s)	KL Divergence↓	NFE	Runtime (s)	KL Divergence↓	NFE
Seq.	/	2.43 ± 0.22	0.0021 ± 0.0005	2200	3.53 ± 0.05	0.0493 ± 0.0038	3480
Par.	2	0.36 ± 0.06	0.0023 ± 0.0004	280	0.77 ± 0.09	0.0500 ± 0.0023	600
	4	0.54 ± 0.11	0.0021 ± 0.0006	360	1.14 ± 0.08	0.0498 ± 0.0047	720
	6	0.70 ± 0.07	0.0019 ± 0.0005	440	1.33 ± 0.09	0.0495 ± 0.0030	840
	8	0.88 ± 0.07	0.0019 ± 0.0004	520	1.89 ± 0.06	0.0491 ± 0.0042	960
	10	1.07 ± 0.10	0.0018 ± 0.0003	600	2.22 ± 0.07	0.0488 ± 0.0043	1080
	12	1.26 ± 0.09	0.0019 ± 0.0004	680	1.59 ± 0.08	0.0482 ± 0.0050	1200
Alg.	K_p	**Hypercube**			**Embedded Hypercube**		
		Runtime (s)	KL Divergence↓	NFE	Runtime (s)	KL Divergence↓	NFE
Seq.	/	3.42 ± 0.07	1.0287 ± 0.0263	5400	3.52 ± 0.13	0.1647 ± 0.0095	6000
Par.	2	1.17 ± 0.06	1.0171 ± 0.0275	500	0.99 ± 0.03	0.1611 ± 0.0105	1200
	4	2.03 ± 0.10	1.0174 ± 0.0296	600	1.18 ± 0.05	0.1582 ± 0.0103	1400
	6	2.88 ± 0.07	1.0078 ± 0.0404	700	1.36 ± 0.05	0.1544 ± 0.0060	1600
	8	3.70 ± 0.05	1.0002 ± 0.0354	800	1.55 ± 0.04	0.1531 ± 0.0083	1800
	10	4.55 ± 0.09	1.0055 ± 0.0271	900	1.77 ± 0.06	0.1552 ± 0.0110	2000
	12	5.42 ± 0.06	0.9982 ± 0.0252	1000	1.89 ± 0.08	0.1564 ± 0.0078	2200

5.1.2 Ablation Study

We conducted an analysis to independently and jointly assess the impact of two distinct algorithmic components—the Predictor and the Corrector—on the final KL divergence on circle and hypercube distributions. The result in Table 4 shows the importance of both correctors and predictors.

Table 4: Ablation study of KL Divergence under different sampling strategies.

Sampling Strategy		**KL Divergence ↓**	
(K_p, K_c)	**Description**	**Ring Distribution**	**Hypercube Distribution**
$(0, 0)$	Baseline	1.6076 ± 0.1551	3.4351 ± 0.1199
$(2, 0)$	Predictor-Only	$0{,}1203 \pm 0.0212$	1.2730 ± 0.0432
$(0, 1)$	Corrector-Only	0.1161 ± 0.0066	0.9388 ± 0.0320
$(2, 1)$	Predictor + Corrector	$\mathbf{0.0628 \pm 0.0038}$	$\mathbf{0.7360 \pm 0.0132}$

5.2 Real-world data experiments

To further test the algorithm's performance on complex high-dimensional real cases, we have added experiments on the acceleration effects in image and text generation from the perspective of the Number of Function Evaluations (NFE). The experiment was conducted on a single NVIDIA RTX4090 laptop GPU with 16GB memory. More experiments about memory cost and hyperparameter are shown in Appendix F.2.

Table 5: Parameter settings and GPU memory cost for image and text generation.

Tasks	**Seq. Memeory Cost**	**Para. Memory Cost**	**M**	**N**	K_p
Text	6.7GB	11.6GB	8	32	2
Image	3.0GB	3.7GB	10	20	2

5.2.1 IMAGE GENERATION

The experiment utilized a MaskGiT-based score model (Chang et al., 2022; Besnier & Chen, 2023) pretrained on ImageNet (Deng et al., 2009). We compared the performance of both sequential and parallel τ-leaping when generating 256×256 resolution images and evaluated the The Fréchet Inception Distance (FID) score based on 10000 samples. The parameter settings are shown in Table 5, and the results are shown in Table 6. Based on the results, when compressing the NFE from 400 to 40, our method still achieves a comparable FID score.

Table 6: Performance of sequential and parallel τ-leaping for image generation.

Imagenet	**NFE**	**Runtime(s)**	**FID**$\downarrow$
Sequential	80	1.92	13.38
Parallel	**40**	**1.43**	**12.05**

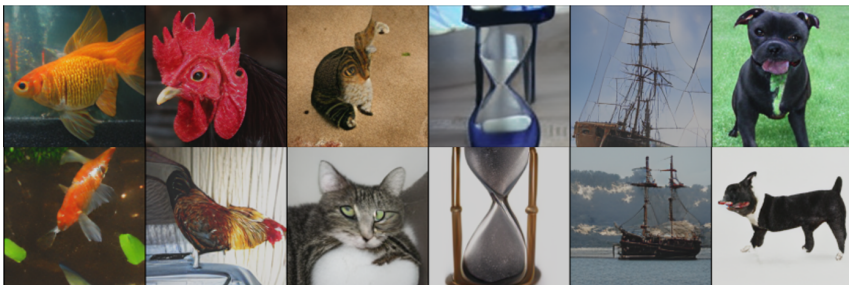

Figure 1: Generated samples from the Imagenet experiments.

5.2.2 TEXT GENERATION

The experiment utilized an RADD-based score model (Ou et al., 2024) pretrained on the OpenWebText dataset (Gokaslan & Cohen, 2019) which has GPT-2-level text generation capabilities (Radford et al., 2019). We compared the performance of both sequential and parallel τ-leaping when generating 512-token texts, and evaluated the average generative perplexity score with 1024 samples. The parameter settings are shown in Table 5, and the results are shown in Table 7. Based on the results, our method achieved a better average perplexity while reducing the NFE from 128 to 64.

Table 7: Performance of sequential and parallel τ-leaping for text generation.

Language Model	**NFE**	**Runtime(s)**	**Avg. Perplexity**$\downarrow$
Sequential	128	2.84	52.698
Parallel	**64**	**2.36**	**48.747**

6 OUR CONCLUSION

In this work, we propose a parallel-in-time τ-leaping algorithm for discrete diffusion models based on Picard iteration within the Continuous-Time Markov Chain (CTMC) and stochastic integral framework. We reduce the time complexity from $\widetilde{\mathcal{O}}(d^2)$ to $\widetilde{\mathcal{O}}(d^{3/2})$ and provide proofs for both computational complexity and error analysis. Furthermore, we introduce MCMC-based predictor-corrector strategies to effectively control the additional error introduced by parallel computation. Our method achieves significant acceleration on various synthetic distribution sampling tasks, providing innovative insights and directions for the future development of discrete diffusion models.

Reproducibility Statement The source code of the experiments is provided in supplemental materials as the form of Jupyter Notebook.

Ethics Statement N/A

REFERENCES

Nima Anari, Sinho Chewi, and Thuy-Duong Vuong. Fast parallel sampling under isoperimetry. *Proceedings of Thirty Seventh Conference on Learning Theory*, pp. 161–185, 2024.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Victor Besnier and Mickael Chen. A pytorch reproduction of masked generative image transformer. *arXiv preprint arXiv:2310.14400*, 2023.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Yang Cao, Daniel T Gillespie, and Linda R Petzold. Efficient step size selection for the tau-leaping simulation method. *The Journal of chemical physics*, 124(4), 2006.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

Chen-Hao Chao, Wei-Fang Sun, Hanwen Liang, Chun-Yi Lee, and Rahul G Krishnan. Beyond masked and unmasked: Discrete diffusion models via partial masking. *arXiv preprint arXiv:2505.18495*, 2025.

Haoxuan Chen, Yinuo Ren, Lexing Ying, and Grant Rotskoff. Accelerating diffusion models with parallel sampling: Inference at sub-linear time complexity. *Advances in Neural Information Processing Systems*, 37:133661–133709, 2024a.

Zixiang Chen, Huizhuo Yuan, Yongqian Li, Yiwen Kou, Junkai Zhang, and Quanquan Gu. Fast sampling via discrete non-markov diffusion models with predetermined transition time. *Advances in Neural Information Processing Systems*, 37:106870–106905, 2024b.

CW Clenshaw. The numerical solution of linear differential equations in Chebyshev series. In *Mathematical Proceedings of the Cambridge Philosophical Society*. Cambridge University Press, 1957.

Vincent Cohen-Addad, David P Woodruff, and Samson Zhou. Streaming Euclidean k-median and k-means with $o(\log n)$ Space. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Oded Goldreich. Computational complexity: a conceptual perspective. *ACM Sigact News*, 2008.

Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein design with guided discrete diffusion. *Advances in neural information processing systems*, 36:12489–12517, 2023.

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10696–10706, 2022.

Shivam Gupta, Linda Cai, and Sitan Chen. Faster diffusion-based sampling with randomized midpoints: Sequential and parallel. *arXiv e-prints*, pp. arXiv–2406, 2024.

Frank P Kelly. *Reversibility and stochastic networks*. Cambridge University Press, 2011.

Michel Ledoux. The geometry of markov diffusion generators. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, 2000.

David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Jose Lezama, Tim Salimans, Lu Jiang, Huiwen Chang, Jonathan Ho, and Irfan Essa. Discrete predictor-corrector diffusion models for image synthesis. In *The Eleventh International Conference on Learning Representations*, 2022.

Yuchen Liang, Yingbin Liang, Lifeng Lai, and Ness Shroff. Discrete diffusion models: Novel analysis and new sampler guarantees. *arXiv preprint arXiv:2509.16756*, 2025.

Samuel Livingstone and Giacomo Zanella. The barker proposal: Combining robustness and efficiency in gradient-based mcmc. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(2):496–523, 2022.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32819–32848, 2024.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Yinuo Ren, Haoxuan Chen, Grant M Rotskoff, and Lexing Ying. How discrete and continuous diffusion meet: Comprehensive analysis of discrete diffusion models via a stochastic integral framework. In *The Thirteenth International Conference on Learning Representations*, 2025.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. *arXiv preprint arXiv:2506.10892*, 2025.

Anirban Sarkar, Yijie Kang, Nirali Somia, Pablo Mantilla, Jessica Lu Zhou, Masayuki Nagai, Ziqi Tang, Chris Zhao, and Peter Koo. Designing dna with tunable regulatory activity using score-entropy discrete diffusion. *bioRxiv*, pp. 2024–05, 2024.

Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky TQ Chen. Flow matching with general discrete paths: A kinetic-optimal perspective. *arXiv preprint arXiv:2412.03487*, 2024.

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:4263–4276, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Tommaso Biancalani, Avantika Lal, Tommi Jaakkola, Sergey Levine, Hanchen Wang, and Aviv Regev. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. *arXiv preprint arXiv:2410.13643*, 2024.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025.

Tianyu Xie, Shuchen Xue, Zijin Feng, Tianyang Hu, Jiacheng Sun, Zhenguo Li, and Cheng Zhang. Variational autoencoding discrete diffusion with enhanced dimensional correlations modeling. *arXiv preprint arXiv:2505.17384*, 2025.

Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1720–1733, 2023.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*, 2024.

Lu Yu and Arnak Dalalyan. Parallelized midpoint randomization for langevin monte carlo. *Stochastic Processes and their Applications*, pp. 104764, 2025.

Leo Zhang. The cosine schedule is fisher-rao-optimal for masked discrete diffusion models. *arXiv preprint arXiv:2508.04884*, 2025.

Lingxiao Zhao, Xueying Ding, Lijun Yu, and Leman Akoglu. Unified discrete diffusion for categorical data. *arXiv preprint arXiv:2402.03701*, 2024.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*, 2023.

Huanjian Zhou and Masashi Sugiyama. Parallel simulation for sampling under isoperimetry and score-based diffusion models. *arXiv preprint arXiv:2412.07435*, 2024.

Huanjian Zhou and Masashi Sugiyama. Parallel simulation for log-concave sampling and score-based diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

Yuanzhi Zhu, Xi Wang, Stephane Lathuiliere, and Vicky Kalogeiton. Dimo: Distilling masked diffusion models into one-step generator. *arXiv preprint arXiv:2503.15457*, 2025.

APPENDICES

A LLM USAGE STATEMENT

In preparing this manuscript, we used large language models (LLM) solely as a polishing tool to improve the clarity and readability of the text. The LLM was not used for generating original ideas, content, or experimental results. All conceptual contributions, analyses, and conclusions presented in this work are entirely from the authors.

B Related Work

B.1 Discrete Diffusion models

Masked and Uniformed discrete diffusion models have developed from a wide range of aspects. Austin et al. (2021) provides foundational formalisms for discrete diffusion—multi-nomial corruption and structured transition matrices as the absorbing states. Chao et al. (2025) introduce intermediate token states between masked/unmasked to avoid redundant computation when sequences barely change across steps. Vignac et al. (2022) performs discrete denoising on graphs by noising/denoising categorical node and edge types. Gu et al. (2022) code sequences with discrete diffusion for text-to-image, improving quality and speed versus auto-regressive token decoders. Gruver et al. (2023) introduces NOS guidance to design protein sequences directly in sequence space, demonstrating antibody optimization in vitro. Wang et al. (2024) pptimizes discrete diffusion generators with task rewards to design biomolecular sequences.

B.2 Acceleration for Discrete Diffusion Sampling

So far, there have been numerous studies on accelerating sampling for discrete diffusion models. Chen et al. (2024b) replaces the standard Markov chain with a non-Markov schedule to skip steps and cut the number of network calls without retraining. Sahoo et al. (2025) adapts consistency distillation to discrete diffusion by constructing the duality connection between continuous and discrete diffusion. Zhu et al. (2025) proposes a token initialization strategy that injects randomness while maintaining similarity to teacher training distribution, achieving one-step distillation of masked diffusion models. Zheng et al. (2023); Ou et al. (2024) design equivalent reparameterization of discrete diffusion that yields more effective training and decoding strategies. Wu et al. (2025) develops confidence-aware parallel decoding to accelerate multi-token sampling while maintaining accuracy. Zhang (2025) gives a principled choice of discretization schedule for efficient sampling. Shaul et al. (2024) allows arbitrary discrete probability paths, giving more control to find shorter or easier trajectories with fewer steps for discrete generation. Zhao et al. (2024) derives a simple backward denoising formula, enabling exact and accelerated sampling and unifying discrete-time and continuous-time discrete diffusion.

B.3 Picard Iteration

Parallel sampling based on Picard recursions has been applied to various models such as first-order Markov chains for both Monte Carlo methods (Yu & Dalalyan, 2025; Anari et al., 2024) and continuous generative models (Zhou & Sugiyama, 2024; Shih et al., 2023; Gupta et al., 2024).

C Backgrounds

In this section, we introduce the basic concepts and existing convergence analysis results of discrete diffusion models in Section 2 and parallel simulation for diffusion model based on Picard iteration in Section C.2.

C.1 Poisson Random Measure

Definition C.1 (Poisson Random Measure). Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(\mathbb{X}, \mathcal{B}, \nu)$ be a measure space satisfying that

$$\int_{\mathbb{X}} 1 \vee |y| \vee |y|^2 \nu(\mathrm{d}y) < \infty,$$

The random measure $N(\mathrm{d}t, \mathrm{d}y)$ on $\mathbb{R}^+ \times \mathbb{X}$ is called a *Poisson random measure* w.r.t. measure ν if it is a random counting measure satisfying the following properties:

(i) For any $B \in \mathcal{B}$ and $0 \leq s < t$, $N((s,t] \times B) \sim \mathcal{P}\left(\nu(B)(t-s)\right)$;

(ii) For any $t \geq 0$ and pairwise disjoint sets $\{B_i\}_{i\in[n]} \subset \mathcal{B}$, $\{N_t(B_i) := N((0,t] \times B_i)\}_{i\in[n]}$ are independent stochastic processes.

Definition C.2 (Poisson Random Measure with Evolving Intensity). Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(\mathbb{X}, \mathcal{B}, \nu)$ be a measure space. Suppose $\lambda_t(y)$ is a non-negative predictable process on $\mathbb{R}^+ \times \mathbb{X} \times \Omega$ satisfying that for any $0 \le T < \overline{T}$,

$$\int_0^T \int_{\mathbb{X}} 1 \vee |y| \vee |y|^2 \lambda_t(y) \nu(\mathrm{d}y)\mathrm{d}t < \infty, \text{ a.s..}$$

The random measure $N[\lambda](\mathrm{d}t, \mathrm{d}y)$ on $\mathbb{R}^+ \times \mathbb{X}$ is called a *Poisson random measure* with *evolving intensity* $\lambda_t(y)$ w.r.t. measure ν if it is a random counting measure satisfying the following properties:

(i) For any $B \in \mathcal{B}$ and $0 \le s < t$, $N[\lambda]((s,t] \times B) \sim \mathcal{P}\left(\int_s^t \int_B \lambda_\tau(y)\nu(\mathrm{d}y)\mathrm{d}\tau\right)$;

(ii) For any $t \ge 0$ and pairwise disjoint sets $\{B_i\}_{i\in[n]} \subset \mathcal{B}$,

$$\{N_t[\lambda](B_i) := N[\lambda]((0,t] \times B_i)\}_{i\in[n]}$$

are independent stochastic processes.

C.2 Continuous Diffusion Models and Picard Iteration

In score-based diffusion models, one considers forward process $(\boldsymbol{x}_t)_{t\in[0,T]}$ in $\mathbb{R}^d$ governed by the canonical Ornstein-Uhlenbeck (OU) process (Ledoux, 2000):

$$\mathrm{d}\boldsymbol{x}_t = -\frac{1}{2}\boldsymbol{x}_t \mathrm{d}t + \mathrm{d}\boldsymbol{B}_t, \qquad \boldsymbol{x}_0 \sim \boldsymbol{q}_0, \qquad t \in [0,T], \tag{11}$$

where $\boldsymbol{q}_0$ is the initial distribution over $\mathbb{R}^d$ and $\boldsymbol{B}_t$ is the Brownian Motion. The corresponding backward process $(\overleftarrow{\boldsymbol{x}}_t)_{t\in[0,T]}$ in $\mathbb{R}^d$ follows an SDE defined as

$$\begin{cases} \mathrm{d}\overleftarrow{\boldsymbol{x}}_t = \left[\frac{1}{2}\overleftarrow{\boldsymbol{x}}_t + \nabla \log \overleftarrow{p}_t(\overleftarrow{\boldsymbol{x}}_t)\right]\mathrm{d}t + \mathrm{d}\boldsymbol{B}_t \quad t \in [0,T], \\ \overleftarrow{\boldsymbol{x}}_0 \sim \boldsymbol{p}_0 \approx \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{I}_d) \end{cases} \tag{12}$$

where $\mathcal{N}(\cdot,\cdot)$ represents the normal distribution over $\mathbb{R}^d$. In practice, the score function $\nabla \log \overleftarrow{p}_t(\overleftarrow{\boldsymbol{x}}_t)$ is estimated by NN $\boldsymbol{s}_t^\theta : \mathbb{R}^d \mapsto \mathbb{R}^d$, where θ is the parameters of NN. The backward process is approximated by

$$\begin{cases} \mathrm{d}\boldsymbol{y}_t = \left[\frac{1}{2}\boldsymbol{y}_t + \boldsymbol{s}_t^\theta(\boldsymbol{y}_t)\right]\mathrm{d}t + \mathrm{d}\boldsymbol{B}_t \quad t \in [0,T], \\ \boldsymbol{y}_0 \sim \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{I}_d). \end{cases} \tag{13}$$

The main idea of parallelized simulation via Picard iteration is to regroup the discrete grids along time horizon and update all grids in same group simultaneously (Clenshaw, 1957; Anari et al., 2024; Zhou & Sugiyama, 2025). Specifically, to approximate the difference $\boldsymbol{x}_{t_{n+1}} - \boldsymbol{x}_{t_n}$ over time slice $[t_n, t_{n+1}]$ as

$$\begin{aligned} \boldsymbol{x}_{t_{n+1}} - \boldsymbol{x}_{t_n} &= \int_{t_n}^{t_{n+1}} \boldsymbol{s}_s^\theta(\boldsymbol{x}_s)\mathrm{d}s + \sqrt{2}(\boldsymbol{B}_{t_{n+1}} - \boldsymbol{B}_{t_n}) \\ &\approx \sum_{i=1}^{M} \boldsymbol{w}_i \boldsymbol{s}_{t_n+\tau_{n,i}}^\theta(\boldsymbol{x}_{t_n+\tau_{n,i}}) + \sqrt{2}(\boldsymbol{B}_{t_{n+1}} - \boldsymbol{B}_{t_n}), \end{aligned}$$

with a discrete grid of M collocation points as $\boldsymbol{x}_{t_n} = \boldsymbol{x}_{t_n+\tau_{n,0}} \le \boldsymbol{x}_{t_n+\tau_{n,1}} \le \boldsymbol{x}_{t_n+\tau_{n,2}} \le \cdots \le \boldsymbol{x}_{t_n+\tau_{n,M}} = \boldsymbol{x}_{t_{n+1}}$. We update the points in a wave-like fashion, which inherently allows for parallelization: for $m' = 1, \ldots, M$,

$$\boldsymbol{x}_{t_n+\tau_{n,m}}^{p+1} = \boldsymbol{x}_{t,n} + \sum_{m=1}^{M-1} \boldsymbol{w}_m \boldsymbol{s}_{t_n+\tau_{n,i}}^\theta(\boldsymbol{x}_{t_n+\tau_{n,m}}^p) + \sqrt{2}(\boldsymbol{B}_{t_n+\tau_{n,m}} - \boldsymbol{B}_{t_n}).$$

With such regrouping, as long as the total time length of each group scales as $\mathcal{O}(1/L)$, the grids will converge exponentially fast. Given a sufficiently accurate starting point at time t_n, the initial error scales as $\mathcal{O}(d)$. Therefore, $K = \mathcal{O}\left(\log\left(\frac{d}{\varepsilon^2}\right)\right)$ steps suffice for the convergence of each group and resulting $\log d/d$ times speed-up.

D MISSING PROOF IN SECTION 3

In this section, we provide the detailed proof about the error bound of Picard-based parallel sampling.

D.1 USEFUL FACTS

Definition D.1 (Mixing Time). We define the mixing time $t_{mix}(\epsilon)$ of the continuous-time Markov chain with rate matrix $\boldsymbol{Q}$ as the smallest time t such that starting from any initial distribution $\boldsymbol{p}_0$, the KL divergence $D_{\mathrm{KL}}(\boldsymbol{p}_t\|\boldsymbol{\pi})$ is less than ϵ, *i.e.*

$$t_{mix}(\epsilon) = \inf\left\{ t \in \mathbb{R}_+ \middle| D_{\mathrm{KL}}(\boldsymbol{p}_t\|\boldsymbol{\pi}) = D_{\mathrm{KL}}(e^{-t\boldsymbol{Q}}\boldsymbol{p}_0\|\boldsymbol{\pi}) \le \epsilon \right\}.$$

Theorem D.1 (Truncation Error Theorem C.1 in Ren et al. (2025)). *The forward process equation 1 converges to the uniform distribution $\boldsymbol{p}_\infty = \mathbf{1}/|\mathbb{X}|$ exponentially fast in terms of the KL divergence,* i.e.

$$D_{\mathrm{KL}}(\boldsymbol{p}_t\|\boldsymbol{p}_\infty) = D_{\mathrm{KL}}\left(\boldsymbol{p}_t \middle\| \frac{\mathbf{1}}{|\mathbb{X}|}\right) \lesssim e^{-\rho t}\log|\mathbb{X}|,$$

where $|\mathbb{X}|$ is the size of the state space, and t_{mix} is the mixing time of the continuous-time Markov chain corresponding to the rate matrix $\boldsymbol{Q}$ defined in Definition D.1.

D.2 STOCHASTIC INTEGRAL FORMULATION OF PARALLEL τ-LEAPING

Proposition D.1 (Stochastic Integral Formulation of backward process (Ren et al., 2025, Proposition 3.2)). *The backward process in discrete diffusion models equation 2 can be represented by the following stochastic integral:*

$$\overleftarrow{x}_t = \overleftarrow{x}_0 + \int_0^t \int_{\mathbb{X}} (y - \overleftarrow{x}_{\tau^-}) N[\mu](\mathrm{d}\tau, \mathrm{d}y), \tag{14}$$

where

$$\mu_\tau(y) = \overleftarrow{s}_\tau(\overleftarrow{x}_{\tau^-}, y)\widetilde{Q}_\tau(\overleftarrow{x}_{\tau^-}, y),$$

and X_{t^-} denotes the left limit of a càdlàg process X_t at time t.

Proposition D.2 (Stochastic Integral Formulation of parallel τ-Leaping). *In time $[t_n, t_{n+1}]$, for any $k \in [K]$, $\widehat{y}_{t_n+m\varepsilon}$ can be interpreted by following stochastic integral equation:*

$$\widehat{y}_t^K = \widehat{y}_{t_n} + \int_0^t \int_{\mathbb{X}} (y - \widehat{y}^{K-1}_{\lfloor\tau\rfloor^-}) N[\widehat{\mu}^{K-1}_{\lfloor\cdot\rfloor}](\mathrm{d}\tau, \mathrm{d}y), \tag{15}$$

where the evolving intensity $\widehat{\mu}^{K-1}_{\lfloor\tau\rfloor}(y)$ is given by

$$\widehat{\mu}^{K-1}_{\lfloor\tau\rfloor}(y) = \overleftarrow{\widehat{s}}_{\lfloor\tau\rfloor}(\widehat{y}^{K-1}_{\lfloor\tau\rfloor^-}, y)\widetilde{Q}_{\lfloor\tau\rfloor}(\widehat{y}^{K-1}_{\lfloor\tau\rfloor^-}, y),$$

in which we used the symbol $\lfloor\tau\rfloor = \min\{t_n + l\epsilon : l \in [N]\}$ for $\tau \in [t_n, t_{n+1})$.

Proof. Without loss of generality, suppose the time block $[t_n, t_{n+1}]$ is uniformly divided into M small steps with grid $(s_i)_{i\in[0:M]}$, then we have

$$\begin{aligned}
\widehat{y}_t^K &= \widehat{y}_{t_n} + \int_0^s \int_{\mathbb{X}} (y - \widehat{y}^{K-1}_{\lfloor\tau\rfloor^-}) N[\widehat{\mu}^{K-1}_{\lfloor\cdot\rfloor}](\mathrm{d}\tau, \mathrm{d}y) \\
&= \widehat{y}_{t_n} + \sum_{i=1}^{\lfloor(t-t_n)/\epsilon\rfloor} \int_{s_{i-1}}^{s_i} \int_{\mathbb{X}} \left(y - \widehat{y}^{K-1}_{s_{i-1}^-}\right) N[\widehat{\mu}^{K-1}_{s_{i-1}}](\mathrm{d}\tau, \mathrm{d}y) \\
&= \widehat{y}_{t_n} + \sum_{i=1}^{\lfloor(t-t_n)/\epsilon\rfloor} \int_{\mathbb{X}} \left(y - \widehat{y}^{K-1}_{s_{i-1}^-}\right) N[\widehat{\mu}^{K-1}_{s_{i-1}}]((s_{i-1}, s_i], \mathrm{d}y),
\end{aligned} \tag{16}$$

since $\mathbb{X}$ is finite, it can be decomposed into the following sum of jumps:

$$\begin{aligned}\widehat{y}_t^K &= \widehat{y}_{t_n} + \sum_{i=1}^{\lfloor (t-t_n)/\epsilon \rfloor} \int_{\mathbb{X}} \left(y - \widehat{y}^{K-1}_{s_{i-1}^-}\right) N[\widehat{\mu}^{K-1}_{s_{i-1}}]((s_{i-1}, s_i], \mathrm{d}y) \\ &= \widehat{y}_{t_n} + \sum_{i=1}^{\lfloor (t-t_n)/\epsilon \rfloor} \sum_{y \in \mathbb{X}} \left(y - \widehat{y}^{K-1}_{s_{i-1}^-}\right) N[\widehat{\mu}^{K-1}_{s_{i-1}}]((s_{i-1}, s_i], \{y\}) \\ &= \widehat{y}_{t_n} + \sum_{i=1}^{\lfloor (t-t_n)/\epsilon \rfloor} \sum_{y \in \mathbb{X}} \left(y - \widehat{y}^{K-1}_{s_{i-1}^-}\right) \mathcal{P}((s_i - s_{i-1})\widehat{\mu}_{s_{i-1}}(y)), \end{aligned} \tag{17}$$

which is exactly the Picard update formula (7) in Section 3.1. □

D.3 Decomposition of KL divergence

Theorem D.2. *Let $\overleftarrow{p}_{0:T}$ and $\widehat{q}_{0:T}$ be the path measures of the backward process with the stochastic integral formulation equation 14 and the interpolating process equation 15 of τ-leaping algorithm, then it holds that*

$$D_{\mathrm{KL}}(\overleftarrow{p}_{0:T} \| \widehat{q}_{0:T}) \leq D_{\mathrm{KL}}(\overleftarrow{p}_0 \| \widehat{q}_0) + \mathbb{E}\left[\int_0^T \int_{\mathbb{X}} \left(\mu_t(y) \log \frac{\mu_t(y)}{\widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)} - \mu_t(y) + \widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)\right) \nu(\mathrm{d}y)\mathrm{d}t\right], \tag{18}$$

where the expectation is taken w.r.t. paths generated by the backward process equation 14.

Proof. By the chain rule of KL divergence, we have

$$D_{KL}(\overleftarrow{p}_{0:T} \| \widehat{q}_{0:T}) = D_{KL}(\overleftarrow{p}_0 \| \widehat{q}_0) + \mathbb{E}[D_{KL}(\overleftarrow{p}_{0:T} \| \widehat{q}_{0:T} | \overleftarrow{x}_0 = y_0 = y)]. \tag{19}$$

One can rewrite the second term as

$$\mathbb{E}[D_{KL}(\overleftarrow{p}_{0:T} \| \widehat{q}_{0:T} | \overleftarrow{x}_0 = y_0 = y)] = \mathbb{E}\left[\log \frac{\mathrm{d}\overleftarrow{p}_{0:T}}{\mathrm{d}\widehat{q}_{0:T}} \mid \overleftarrow{x}_0 = y_0 = y)\right] = \mathbb{E}\left[\log Z_T^{-1}\left[\frac{\widehat{\mu}^{K-1}}{\mu}\right]\right]. \tag{20}$$

where Z_T denotes the Change of Measure in Theorem 3.3 in (Ren et al., 2025) with the form of

$$\log Z_T[h] = \int_0^t \int_{\mathbb{X}} \log h_\tau(y) N[\lambda](\mathrm{d}\tau \times \mathrm{d}y) - \int_0^t \int_{\mathbb{X}} (h_\tau(y) - 1)\lambda_\tau(y)\nu(\mathrm{d}y)\mathrm{d}\tau. \tag{21}$$

Given $h = \frac{\widehat{\mu}^{K-1}}{\mu}$, we have

$$\begin{aligned}\log\left(Z_T\left[\frac{\widehat{\mu}^{K-1}}{\mu}\right]\right) &= \int_0^T \int_{\mathbb{X}} \log\left(\frac{\widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)}{\mu_t(y)}\right) N[\mu](\mathrm{d}t \times \mathrm{d}y) \\ &- \int_0^T \int_{\mathbb{X}} \left(\frac{\widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)}{\mu_t(y)} - 1\right) \mu_s(y)\nu(\mathrm{d}y)\mathrm{d}t\end{aligned}$$

Eventually, we can compute the expectation as

$$\begin{aligned}\mathbb{E}\left[\log Z_T^{-1}\left[\frac{\widehat{\mu}^{K-1}}{\mu}\right]\right] &= \mathbb{E}\left[\int_0^T \int_{\mathbb{X}} \left(-\log\left(\frac{\widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)}{\mu_t(y)}\right) + \frac{\widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)}{\mu_t(y)} - 1\right) \mu_t \nu(\mathrm{d}y)\mathrm{d}t\right] \\ &= \mathbb{E}\left[\int_0^T \int_{\mathbb{X}} \left(\mu_t(y) \log \frac{\mu_t(y)}{\widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)} - \mu_t(y) + \widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)\right) \nu(\mathrm{d}y)\mathrm{d}t\right]\end{aligned}$$

□

Now, the problem remaining is reduced to bound the following discrepancy with $G(x;y) = x(\log x - \log y) - x$,

$$\begin{aligned}
&\int_{t_n}^{t_{n+1}} \int_{\mathbb{X}} \left(\mu_t(y) \log \frac{\mu_t(y)}{\widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)} - \mu_t(y) + \widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y) \right) \nu(\mathrm{d}y)\mathrm{d}t \\
&= \int_{t_n}^{t_{n+1}} \int_{\mathbb{X}} \Bigg(\underbrace{\mu_{\lfloor t \rfloor}(y) \log \frac{\mu_{\lfloor t \rfloor}(y)}{\widehat{\mu}^{K}_{\lfloor t \rfloor}(y)} - \mu_t(y) + \widehat{\mu}^{K}_{\lfloor t \rfloor}(y)}_{:=A_{t_n}} \\
&\quad + \underbrace{G(\mu_t(y); \widehat{\mu}^{K}_{\lfloor t \rfloor}(y)) - G(\mu_{\lfloor t \rfloor}(y); \widehat{\mu}^{K}_{\lfloor t \rfloor}(y))}_{:=B_{t_n}} \\
&\quad + \underbrace{\mu_t(y) \log \frac{\widehat{\mu}^{K}_{\lfloor t \rfloor}(y)}{\widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)} - \widehat{\mu}^{K}_{\lfloor t \rfloor}(y) + \widehat{\mu}^{K-1}_{\lfloor t \rfloor}(y)}_{:=C_{t_n}} \Bigg) \nu(\mathrm{d}y)\mathrm{d}s;
\end{aligned}$$

where A_{t_n} is the estimation error of score function, B_{t_n} measures the discretization error, and C_{t_n} measures the error by Picard iteration.

D.4 Discretization Error and Estimation Error

Proposition D.3 (Discretization error and Estimation Error (Ren et al., 2025)). *For any $n \in [N]$, we have*

$$\int_0^T \int_{\mathbb{X}} A_{t_n} \nu(\mathrm{d}y)\mathrm{d}s \le \delta^2 \quad \textit{and} \quad \int_{t_n}^{t_{n+1}} \int_{\mathbb{X}} B_{t_n} \nu(\mathrm{d}y)\mathrm{d}s \le \overline{D}^2 \epsilon h.$$

D.5 Convergence of Picard iteration

Proposition D.4 (Convergence of Picard iteration). *By taking $h = m\epsilon = \mathcal{O}(\frac{1}{d^{3/2}})$, for any $n \in [N]$, $j \in [K]$, $y \in \mathbb{X}$ and $t \in [t_n, t_{n+1}]$, we have*

$$\max_{t \in [t_n, t_{n+1}]} \mathbb{E} \left\| \widehat{y}^{K-j}_{\lfloor t \rfloor} - \widehat{y}^{K-j-1}_{\lfloor t \rfloor} \right\| \le C \max_{t \in [t_n, t_{n+1}]} \mathbb{E} \left\| \widehat{y}^{K-j-1}_{\lfloor t \rfloor} - \widehat{y}^{K-j-2}_{\lfloor t \rfloor} \right\|.$$

with some constant $C \in (0,1)$.

Proof. The update step inside the block is as follows:

$$\widehat{y}^{(k+1)}_m = \widehat{y}^{(k+1)}_0 + \sum_{j=0}^{m-1} \Delta \widehat{y}^{(k)}_j$$

where

$$\Delta \widehat{y}^{(k)}_j = \sum_{y' \in \mathbb{X}} (y' - \widehat{y}^{(k)}_{t_n + j\epsilon}) \cdot \mathrm{Numjump}^{(k)}_j(y'), \quad \mathrm{Numjump}^{(k)}_j(y') \sim \mathcal{P}(\widehat{\mu}^{k}_{t_n+j\epsilon}(y' | \widehat{y}^{(k)}_{t_n+j\epsilon}) \cdot \epsilon).$$

Suppose the state difference between iteration $k+1$ and k in block n at time $t_n + m\epsilon$ is defined as

$$\epsilon_{k+1}(m) = \widehat{y}^{(k+1)}_{t_n+m\epsilon} - \widehat{y}^{(k)}_{t_n+m\epsilon},$$

such difference can also be considered as the difference of accumulated jump:

$$\epsilon_k(m) = \sum_{j=0}^{m-1} \left(\Delta \widehat{y}^{(k)}_j - \Delta \widehat{y}^{(k-1)}_j \right).$$

Take square norm on both sides and use Cauchy-Schwarz Inequality, we have

$$\mathbb{E}[\|\epsilon_k(m)\|^2] = \mathbb{E}\left[\left\| \sum_{j=0}^{m-1} \left(\Delta \widehat{y}^{(k)}_j - \Delta \widehat{y}^{(k-1)}_j \right) \right\|^2 \right] \le m \sum_{j=0}^{m-1} \mathbb{E}\left[\left\| \Delta \widehat{y}^{(k)}_j - \Delta \widehat{y}^{(k-1)}_j \right\|^2 \right].$$

Now we split the difference between two total jump vectors by using $\|A+B\|^2 \le 2\|A\|^2 + 2\|B\|^2$:

$$\mathbb{E}\left[\left\|\Delta\widehat{y}_j^{(k)} - \Delta\widehat{y}_j^{(k-1)}\right\|^2\right] \le \underbrace{2\mathbb{E}\left[\left\|\sum_{y'\in\mathbb{X}}(y'-\widehat{y}_j^{(k)})(\mathrm{NumJump}_j^{(k)}(y') - \mathrm{NumJump}_j^{(k-1)}(y'))\right\|^2\right]}_{I_1}$$
$$+ \underbrace{2\mathbb{E}\left[\left\|(\widehat{y}_j^{(k-1)} - \widehat{y}_j^{(k)})\sum_{y'\in\mathbb{X}}\mathrm{NumJump}_j^{(k-1)}(y')\right\|^2\right]}_{I_2}$$

For the first term, from the Cauchy-Schwarz Inequality, we have:

$$I_1 \le \|\mathbb{X}\|\sum_{y'\in\mathbb{X}}\mathbb{E}\Big[\|y'-\widehat{y}_j^{(k)}\|^2\cdot(\mathrm{NumJump}_j^{(k)}(y') - \mathrm{NumJump}_j^{(k-1)}(y'))^2\Big]$$

We define $R_{\mathbb{X}}^2 = \max_{x_1,x_2\in\mathbb{X}}\|x_1-x_2\|^2$ as the scale of the state space. From the Lipschitz Continuity in Assumption 3.4, we have

$$I_1 \le \|\mathbb{X}\|\cdot R_{\mathbb{X}}^2\cdot(L_\mu\epsilon)^2\mathbb{E}[\|\widehat{y}_j^{(k)} - \widehat{y}_j^{(k-1)}\|^2]$$

For the second, we utilize the Law of total expectation to decompose the form as:

$$\begin{aligned} I_2 &= \mathbb{E}\left[\mathbb{E}\left[\left\|\widehat{y}_j^{(k-1)} - \widehat{y}_j^{(k)}\right\|^2\left(\sum_{y'\in\mathbb{X}}\mathrm{NumJumps}_j^{(k-1)}(y')\right)^2\middle|\widehat{y}_j^{(k-1)},\widehat{y}_j^{(k)}\right]\right] \\ &= \mathbb{E}\left[\left\|\widehat{y}_j^{(k-1)} - \widehat{y}_j^{(k)}\right\|^2\mathbb{E}\left[\left(\sum_{y'\in\mathbb{X}}\mathrm{NumJump}_j^{(k-1)}(y')\right)^2\middle|\widehat{y}_j^{(k-1)}\right]\right] \end{aligned} \tag{22}$$

For the inner expectation, the sum of NumJump is also the sum of independent Poisson variables, which is still a Poisson variable. Thus we have

$$\lambda_{total} = \sum_{y'\in\mathbb{X}}\mathbb{E}\big[\mathrm{NumJump}_j^{(k-1)}(y')|\widehat{y}_j^{(k-1)}\big] = \sum_{y'\in\mathbb{X}}\mu_j^\theta(y'|\widehat{y}_j^{(k-1)})\cdot\epsilon$$

According to the assumption of rate matrix regularization and bounded score, we have

$$\sum_{y'\neq\widehat{y}_j^{(k-1)}}\mu_j^\theta(y'|\widehat{y}_j^{(k-1)}) = \sum_{y'\neq\widehat{y}_j^{(k-1)}}\widehat{s}^\theta(\widehat{y}_j^{(k-1)},y')\widetilde{Q}(\widehat{y}_j^{(k-1)},y') \le M_s\bar{D} := \Lambda_{max}.$$

For $x\sim\mathcal{P}(\lambda)$, the second momentun is $\mathbb{E}[x^2] = \lambda^2+\lambda$, therefore we have

$$\mathbb{E}\left[\left(\sum_{y'\in\mathbb{X}}\mathrm{NumJump}_j^{(k-1)}(y')\right)^2\middle|\widehat{y}_j^{(k-1)}\right] \le (\epsilon\Lambda_{max})^2 + (\epsilon\Lambda_{max}),$$

and the second term is bounded as

$$I_2 \le 2\mathbb{E}\big[\|\widehat{y}_j^{(k-1)} - \widehat{y}_j^{(k)}\|^2\big]\cdot(\epsilon^2\Lambda_{max}^2 + \epsilon\Lambda_{max}). \tag{23}$$

Overall, we have

$$\begin{aligned} \mathbb{E}\left[\left\|\Delta\widehat{y}_j^{(k)} - \Delta\widehat{y}_j^{(k-1)}\right\|^2\right] &\le (\|\mathbb{X}\|\cdot R_{\mathbb{X}}^2\cdot(L_\mu\epsilon)^2 + \epsilon^2\Lambda_{max}^2 + \epsilon\Lambda_{max})\cdot\mathbb{E}\big[\|\widehat{y}_j^{(k-1)} - \widehat{y}_j^{(k)}\|^2\big] \\ &= C_0\cdot\mathbb{E}\big[\|\widehat{y}_j^{(k-1)} - \widehat{y}_j^{(k)}\|^2\big], \end{aligned} \tag{24}$$

with $C_0 = \|\mathbb{X}\|\cdot R_{\mathbb{X}}^2\cdot(L_\mu\epsilon)^2 + \epsilon^2\Lambda_{max}^2 + \epsilon\Lambda_{max}$. Thus, we have

$$\begin{aligned} \max_{t\in[t_n,t_{n+1}]}\mathbb{E}\left\|\widehat{y}_{\lfloor t\rfloor}^{K-j} - \widehat{y}_{\lfloor t\rfloor}^{K-j-1}\right\| &\le m\sum_{j=0}^{m-1}\mathbb{E}\left[\left\|\Delta\widehat{y}_j^{K-j-1} - \Delta\widehat{y}_j^{K-j-2}\right\|^2\right] \\ &\le m\cdot C_0\cdot\sum_{j=0}^{m-1}\mathbb{E}\big[\|\widehat{y}_j^{K-j-1} - \widehat{y}_j^{K-j-2}\|^2\big] \qquad (25) \\ &\le m^2C_0\mathbb{E}\big[\|\widehat{y}_j^{K-j-1} - \widehat{y}_j^{K-j-2}\|^2\big] \qquad (26) \\ &\le C\max_{t\in[t_n,t_{n+1}]}\mathbb{E}\left\|\widehat{y}_{\lfloor t\rfloor}^{K} - \widehat{y}_{\lfloor t\rfloor}^{K-j}\right\|, \qquad (27) \end{aligned}$$

the last inequality holds if we take $h = m\epsilon = \mathcal{O}(\frac{\delta}{d^{3/2}})$. □

D.6 Proof of Theorem 3.1

Combining Theorem D.1, Lemma D.2, Proposition D.3, and Proposition D.4, we have

$$\begin{aligned} D_{\mathrm{KL}}(\overleftarrow{p}_{0:T}\|\widehat{q}_{0:T}) &\le D_{\mathrm{KL}}(\overleftarrow{p}_0\|\widehat{q}_0) + \mathbb{E}\left[\int_0^T \int_{\mathbb{X}} \left(\mu_t(y)\log\frac{\mu_t(y)}{\widehat{\mu}^{K-1}_{\lfloor t\rfloor}(y)} - \mu_t(y) + \widehat{\mu}^{K-1}_{\lfloor t\rfloor}(y)\right)\nu(\mathrm{d}y)\mathrm{d}t\right] \\ &\le D_{\mathrm{KL}}(\overleftarrow{p}_0\|\widehat{q}_0) + \sum_{n=1}^{N}\mathbb{E}\int_{t_n}^{t_{n+1}}\int_{\mathbb{X}} At_n + B_{t_n} + C_{t_n}\nu(\mathrm{d}y)\mathrm{d}s \\ &\le \lesssim e^{-\rho T}\log|\mathbb{X}| + \delta^2 + \overline{D}^2\epsilon T + C^K\mathrm{poly}(\overline{D}). \end{aligned}$$

E Missing Proof in Section 4

E.1 Proof of Theorem 4.1: Analysis of Algorithm 2

Suppose $\bar{p}_{t_n}$ is the true target distribution derived from the forward process, and $\widehat{q}_{t_n}$ is the generated distribution from our predictor-corrector algorithm. We use the total variance distance $E_n = TV(\widehat{q}_{t_n}, \bar{p}_{t_n})$ as the measurement of the error before entering the n-th block computation.

The predictor generates $\widehat{q}_{\mathrm{pred}}$ from the initial distribution $\widehat{q}_{t_n}$. Suppose the predictor also generates a $\widetilde{q}_{\mathrm{pred}}$ from the true distribution $\bar{p}_{t_n}$, then the error generated by the predictor $TV(\widehat{q}_{\mathrm{pred}}, \bar{p}_{t_{n+1}})$ can be decomposed by the trigonometric inequality as

$$TV(\widehat{q}_{\mathrm{pred}}, \bar{p}_{t_{n+1}}) \le TV(\widehat{q}_{\mathrm{pred}}, \widetilde{q}_{\mathrm{pred}}) + TV(\widetilde{q}_{\mathrm{pred}}, \bar{p}_{t_{n+1}}). \tag{28}$$

The first term can be considered as the propagation of the error. Since $\widehat{q}_{\mathrm{pred}}$ and $\widetilde{q}_{\mathrm{pred}}$ are generated by the same mapping (parallel τ-leaping) from the initial $(\widehat{q}_{t_n}, \bar{p}_{t_n})$, according to the Data Processing Inequality [], we have

$$TV(\widehat{q}_{\mathrm{pred}}, \widetilde{q}_{\mathrm{pred}}) \le TV(\widehat{q}_{t_n}, \bar{p}_{t_n}) = E_n, \tag{29}$$

which means the predictor will not amplify the previous error.

The second term $TV(\widetilde{q}_{\mathrm{pred}}, \bar{p}_{t_{n+1}})$ means the new error introduced by the predictor, which consists of the discretization error from τ-leaping and Picard iteration, and the approximation error from the network learning. We denote this term as Δ_p. Therefore the bound of the total error after the predictor stage is:

$$E_{\mathrm{pred}} \le E_n + \Delta_p. \tag{30}$$

The corrector use $\widehat{q}_{\mathrm{pred}}$ as the initial distribution, and generate $\widehat{q}_{t_{n+1}}$ after running K_c steps of Metropolis-Hastings MCMC process, which is designed with the target distribution $\bar{p}_{t_{n+1}}$ as the only stationary distribution. Since the transfer kernel of the MCMC is a contraction mapping with respect to the Total Variance distance with the factor γ, after K_c steps, we have

$$E_{n+1} = TV(\widehat{q}_{t_{n+1}}, \bar{p}_{t_{n+1}}) \le \gamma^{K_c} E_{\mathrm{pred}} + \Delta_c = \gamma^{K_c}(E_n + \Delta_p) + \Delta_c, \tag{31}$$

where Δ_c is the small approximation error of the corrector itself, since it also used $\widehat{s}^{\theta}_t$ for computing acceptance rate. Eventually, the total error after N blocks can be computed as

$$E_N \le \frac{\gamma^{K_c}\Delta_p + \Delta_c}{1 - \gamma^{K_c}} + \gamma^{NK_c}E_0 \tag{32}$$

The estimation of the contract factor γ is described in the following Proposition.

Proposition E.1 (The Estimation of MCMC Contraction)**.** *In finite state space Ω, suppose there is a reversible MCMC transfer kernel P with non-negative eigenvalues and stationary distribution π. Denote the second largest eigenvalues of P as λ_* and Φ as the conductance of the graph corresponding to P, for any distribution μ, we have*

$$\|\mu P^t - \pi\|_{TV} \le \frac{1}{2}\sqrt{\frac{1}{\pi_{\min}} - 1}\left(1 - \frac{\Phi^2}{2}\right)^t \tag{33}$$

Proof. Let $f := \frac{d\mu}{d\pi}$ so that $\mu = f\,\pi$ and $\mu - \pi = (f-1)\pi$. First, by Cauchy–Schwarz,

$$\|\mu - \pi\|_{\mathrm{TV}} = \frac{1}{2}\sum_x \pi(x)\,|f(x)-1| = \frac{1}{2}\|f-1\|_{1,\pi} \le \frac{1}{2}\|f-1\|_{2,\pi}.$$

Reversibility implies P is self-adjoint on $L^2(\pi)$ and (using detailed balance)

$$\frac{d(\mu P^t)}{d\pi}(y) = \sum_x f(x)\,\frac{\pi(x)}{\pi(y)}\,P^t(x,y) = \sum_x f(x)\,P^t(y,x) = (P^t f)(y),$$

hence $\mu P^t - \pi = \big(P^t(f-1)\big)\pi$ and therefore

$$\|\mu P^t - \pi\|_{\mathrm{TV}} = \tfrac{1}{2}\|P^t(f-1)\|_{1,\pi} \le \tfrac{1}{2}\|P^t(f-1)\|_{2,\pi}. \tag{34}$$

Next, since the chain is reversible, the spectrum of P lies in $[0,1]$ and there is an orthonormal eigenbasis of $L^2_0(\pi) := \{g : \sum_x g(x)\pi(x) = 0\}$. Expanding $g \in L^2_0(\pi)$ in that basis gives

$$\|P^t g\|_{2,\pi} \;\le\; \lambda_*^t\,\|g\|_{2,\pi} = (1-\gamma)^t\|g\|_{2,\pi}.$$

Applying this with $g = f - 1$ and combining with (equation 34) yields

$$\|\mu P^t - \pi\|_{\mathrm{TV}} \;\le\; \tfrac{1}{2}\,(1-\gamma)^t\,\|f-1\|_{2,\pi}. \tag{35}$$

We now bound the initial L^2 deviation:

$$\|f-1\|^2_{2,\pi} = \sum_x \pi(x)\Big(\frac{\mu(x)}{\pi(x)} - 1\Big)^2 = \sum_x \frac{\mu(x)^2}{\pi(x)} - 1 \le \frac{1}{\pi_{\min}}\sum_x \mu(x)^2 - 1 \le \frac{1}{\pi_{\min}} - 1,$$

where we used $\pi(x) \ge \pi_{\min}$ and $\mu(x)^2 \le \mu(x)$ for probabilities. Hence

$$\|f-1\|_{2,\pi} \;\le\; \sqrt{\frac{1}{\pi_{\min}} - 1}. \tag{36}$$

Finally, Cheeger's inequality for lazy reversible chains states

$$\gamma \;\ge\; \frac{\Phi^2}{2},$$

so $(1-\gamma)^t \le \big(1 - \frac{\Phi^2}{2}\big)^t$. Plugging this and (equation 36) into (equation 35) gives

$$\|\mu P^t - \pi\|_{\mathrm{TV}} \;\le\; \frac{1}{2}\sqrt{\frac{1}{\pi_{\min}} - 1}\,\Big(1 - \frac{\Phi^2}{2}\Big)^t.$$

□

E.2 Proof of Proposition 4.2: Analysis of Algorithm 3

Fix $x \in \mathbb{X}$ and a token b with $b \sim q_i(\cdot|x) > 0$, and set $x' = x^{(i\leftarrow)b}$ and $a = x_i$. Use the Barker acceptance with the standard proposal-ratio correction:

$$\alpha_i(x \to x') = \begin{cases} \dfrac{r_i(x,x')}{1 + r_i(x,x')}, & \text{if } q_i(a|x') > 0, \\ 0, & \text{otherwise,} \end{cases} \qquad \text{where} \quad r_i(x,x') \;=\; \frac{\pi_t(x')\,q_i(a|x')}{\pi_t(x)\,q_i(b|x)}\,.$$

Setting $\alpha_i = 0$ when the reverse proposal has zero support will enforce symmetric support and avoids irreversibility. Such a setting defines a single-coordinate Markov kernel K_i as

$$K_i(x,x') = \begin{cases} q_i(b|x)\,\alpha_i(x \to x'), & \text{if } x' = x^{(i\leftarrow b)} \neq x, \\ 1 \;-\; \displaystyle\sum_b q_i(b|x)\,\alpha_i\big(x \to x^{(i\leftarrow b)}\big), & \text{if } x' = x, \\ 0, & \text{others.} \end{cases}$$

We first prove two following lemma.

Lemma E.1 (Anchor Cancellation)**.** For any x, i, $\widetilde{x}^{(i)} \equiv x^{(i\leftarrow M')}$, and $a, b \in \mathcal{V}$, we have

$$\log \frac{\pi_t\big(x^{(i\leftarrow b)}\big)}{\pi_t(x)} = \log \underbrace{\frac{\pi_t\big(x^{(i\leftarrow b)}\big)}{\pi_t\big(x^{(i\leftarrow M')}\big)}}_{u_{i,b}(\widetilde{x}^{(i)})} - \log \underbrace{\frac{\pi_t(x)}{\pi_t\big(x^{(i\leftarrow M')}\big)}}_{u_{i,a}(\widetilde{x}^{(i)})} = u_{i,b}(\widetilde{x}^{(i)}) - u_{i,a}(\widetilde{x}^{(i)}).$$

Lemma E.2 (Proposal on Anchored Context)**.** If $q_i(\cdot|x)$ is defined on $\mathcal{K}_i(\widetilde{x}^{(i)})$, then for $x' = x^{(i\leftarrow b)}$, we have

$$q_i(\cdot|x') \equiv q_i(\cdot|x)$$

whenever both are defined, because $\widetilde{x}^{(i)}$ is identical for x and x'. In particular, if q_i is *uniform* on a fixed-size set $\mathcal{K}_i$, then

$$q_i(b|x) = q_i(a|x') = \frac{1}{K}$$

whenever both are positive.

Now we prove the main result. Consider any $x' \neq x$ such that $K_i(x, x') > 0$; then $x' = x^{(i\leftarrow b)}$ for some b with $q_i(b|x) > 0$. Let $a = x_i$ again. If $q_i(a|x') = 0$, our definition sets $\alpha_i(x \to x') = 0$, hence $K_i(x, x') = 0$ and detailed balance holds trivially because $K_i(x', x) = 0$ as well since the reverse move is never proposed.

Otherwise $q_i(a|x') > 0$. By the acceptance rule,

$$\pi_t(x)\, K_i(x, x') = \pi_t(x)\, q_i(b|x)\, \frac{r_i(x, x')}{1 + r_i(x, x')},$$
$$\pi_t(x')\, K_i(x', x) = \pi_t(x')\, q_i(a|x')\, \frac{1}{1 + r_i(x, x')},$$

since the reverse acceptance uses $r_i(x', x) = 1/r_i(x, x')$ under Barker. Taking the ratio,

$$\frac{\pi_t(x)\, K_i(x, x')}{\pi_t(x')\, K_i(x', x)} = \frac{\pi_t(x)\, q_i(b|x)\, \frac{r_i}{1+r_i}}{\pi_t(x')\, q_i(a|x')\, \frac{1}{1+r_i}} = \frac{\pi_t(x)\, q_i(b|x)\, r_i(x, x')}{\pi_t(x')\, q_i(a|x')} = 1,$$

by the very definition

$$r_i(x, x') \;=\; \frac{\pi_t(x')\, q_i(a|x')}{\pi_t(x)\, q_i(b|x)}\,.$$

Hence $\pi_t(x)\, K_i(x, x') = \pi_t(x')\, K_i(x', x)$ for all off-diagonal pairs with positive mass. Including the diagonal case, which follows by summing both sides over x', detailed balance condition holds. Therefore K_i is π_t-reversible and, in particular, π_t-invariant.

F Experiments

We provide extra experiment information in this section.

F.1 Experiment Description

F.1.1 Chessboard

The chessboard distribution has several key characteristics that make it an excellent test benchmark: (1) Discrete: The distribution is defined on a finite set of points on a two-dimensional grid; (2) Sparse: Nearly half of the grid points have a probability of exactly zero. A successful sampler must learn to restrict its generated samples strictly to the points that have non-zero probability mass (i.e., the support of the distribution); (3) Multi-modal: The probability mass is distributed across multiple, disconnected modes rather than being concentrated in a single region. The sampler is required to capture all of these distinct modes; (4) Structured: It exhibits a distinct, non-random geometric structure. This challenges the sampler's ability to reproduce the correct global pattern, rather than merely matching general statistical moments.

We conduct the experiment on a 8×8 chessboard distribution with varying Picard iteration depths K_p. We fix $N = 40, M = 50$ and corrector step $K_c = 5$ with totally 8196 samples. Runtimes and KL Divergence are averaged over 20 runs.

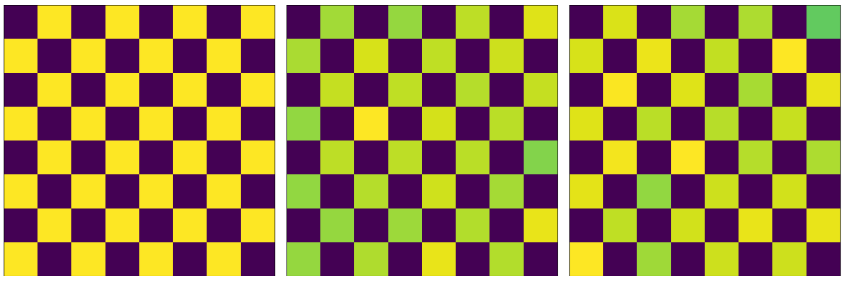

Figure 2: Visualization of the chessboard experiments. (Left) The target distribution. (Middle) The Picard sampling result. (Right) The sequential sampling result.

F.1.2 CIRCLE

The ring distribution on a 2D discrete grid concentrates its entire probability mass on grid points located within an annulus defined by an inner radius r_{in} and an outer radius r_{out}. A key characteristic is its Non-Convexity; the high-probability region encloses a central "hole" of zero probability, which leads to the distribution's support non-convex. Another defining feature is its Connectivity. Unlike the disjoint, multi-modal structure of the checkerboard distribution, the support of the ring distribution forms a single connected component, meaning any point on the ring can traverse to any other point through a series of steps to adjacent locations.

We conduct the experiment on a circle distribution at 32×32 2D grid. with varying Picard iteration depths K_p. We fix $N = 60, M = 50$ and corrector step $K_c = 8$ with totally 4096 samples. Runtime and KL Divergence are averaged over 20 runs.

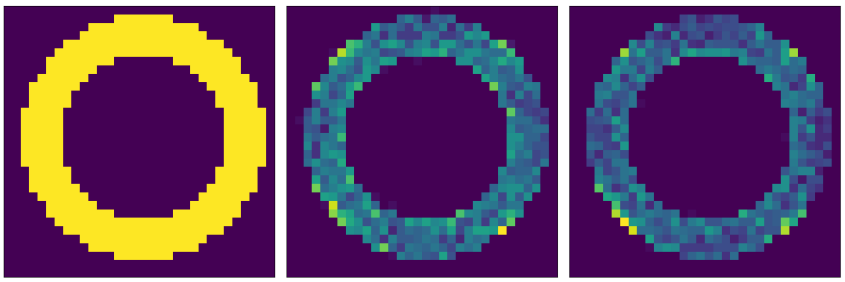

Figure 3: Visualization of the circle experiments. (Left) The target distribution. (Middle) The Picard sampling result. (Right) The sequential sampling result.

F.1.3 HYPERCUBE

The d-dimensional hypercube distribution is a uniform distribution over2^d vertices of a hypercube, where each vertex corresponds to a unique binary vector in $\{0, 1\}^d$. With its well-defined graph structure, exponential state space, and large diameter, this distribution provides a rigorous benchmark for evaluating the high-dimensional scalability and mixing efficiency of sampling algorithms.

We conduct the experiment on a 6-dimensional hypercube distribution. We fix $N = 100, M = 50$ and corrector step $K_c = 10$ with totally 2048 samples. Runtime and KL Divergence are averaged over 20 runs.

F.1.4 EMBEDDED HYPERCUBE

We design an experiment with a controlled intrinsic dimension by embedding a k-dimensional subcube within a d-dimensional ambient space $k < d$. While the ambient space is the full hypercube, the probability mass is confined to a uniform distribution on the subcube. This is achieved by fixing the final $d - k$ bits to 0. The objective is to test the sampler's ability to identify the low-

dimensional manifold containing the data support. A proficient sampler should learn to set the last $d - k$ dimensions to 0 and generate uniform bit combinations for the first k dimensions.

We conduct the experiment on a 6-dimensional subcube distribution embedded in a 12-dimensional hypercube. We fix $N = 50, M = 100$ and corrector step $K_c = 8$ with totally 2048 samples. Runtime and KL Divergence are averaged over 20 runs.

F.2 Computational Cost Analysis

To demonstrate the scalability of our algorithm with respect to data dimension d, we conducted additional experiments on sampling efficiency using d-dimensional independent Bernoulli distributions ($N = 40, M = 10, K_p = 3$). We observed that as d increases from 32 to 2048, the accuracy remains stable.

Table 8: Performance of sequential and parallel tau-leaping with different data dimension d.

Dim	Sequential (s)	Parallel (s)	Sequential Error	Parallel Error	GPU Memory (MB)
8	0.4341	0.0808	0.0042	0.0026	326
16	0.4344	0.0805	0.0037	0.0045	341
32	0.4242	0.0813	0.0049	0.0047	359
64	0.4155	0.0837	0.0035	0.0031	395
128	0.4153	0.0882	0.0038	0.0025	469
256	0.4203	0.1235	0.0044	0.0053	614
512	0.4097	0.2202	0.0040	0.0034	906
1024	0.5463	0.4144	0.0038	0.0033	1487
2048	1.1652	0.8041	0.0046	0.0042	2647

Although the runtime of our parallel algorithm begins to rise noticeably after $d = 128$, it maintains a performance advantage over the serial algorithm. As d increases further, the runtime of the serial algorithm spikes significantly after $d = 1024$, with a rate of growth that eventually surpasses that of the parallel approach. This empirically validates that, despite the additional logarithmic increase in complexity, the reduction in the polynomial term ensures the parallel algorithm remains advantageous in large d cases.

To demonstrate the scalability with respect to the number of small steps M, we tested on 256-dimensional independent Bernoulli distributions sampling with M from 10 to 40. ($N = 10, K_p = 3$)

Table 9: Performance of sequential and parallel tau-leaping with different data dimension M.

M	Parallel (s)	Parallel Error	GPU Memory (MB)
10	0.1235	0.0053	614
20	0.2668	0.0048	894
30	0.3610	0.0029	1176
40	0.4572	0.0031	1481