MCMC-GUIDED REMASKING FOR DISCRETE DIFFUSION: BIAS-CONTROLLED PARALLEL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion models (MDMs) offer a promising alternative to autoregressive language models by enabling parallel token generation. Recent self-correction techniques, such as remasking, can improve generation quality, but their behavior under parallel token updates still remains poorly understood. In particular, the induced transition kernels can deviate from the desired sampling dynamics, undermining the motivation of diffusion-based generation. In this work, we introduce **BReSC** (**B**alanced **Re**masking with **S**taleness **C**ontrol), a training-free remasking sampler grounded in a principled Markov chain Monte Carlo (MCMC) viewpoint. We identify two fundamental biases in parallel remasking: drift bias caused by unbalanced transitions, and a staleness bias caused by simultaneously updating coupled tokens from stale contexts. BReSC addresses these biases through an entropy-balanced remasking policy satisfying coordinate-wise detailed balance, together with an online optimization routine that reduces a staleness proxy by suppressing high-interference parallel updates. We theoretically characterize both biases under idealized assumptions, guiding the design of our practical sampler. We benchmark BReSC on OpenWebText unconditional generation and downstream LLaDA tasks against existing baselines.

1 INTRODUCTION

Diffusion models have gained significant traction for their ability to generate high-quality continuous data such as images and videos (Ho et al., 2020; Song & Ermon, 2019). Recently, discrete counterparts of diffusion models (Austin et al., 2021; Campbell et al., 2022; Chang et al., 2022; Gong et al., 2022; Li et al., 2022; Lou et al., 2023; Sahoo et al., 2024) have emerged as a compelling alternative to autoregressive (AR) transformers for language generation (Chang et al., 2022; Gong et al., 2022; Li et al., 2022; Sahoo et al., 2024), promising to accelerate text generation via parallel decoding. In particular, masked diffusion models (MDMs) (Chang et al., 2022; Nie et al., 2025; Sahoo et al., 2024; Shi et al., 2024; Ye et al., 2025) have become the standard in this space. Unlike AR models that generate tokens strictly left-to-right, MDMs predict multiple tokens simultaneously, leveraging a special `[MASK]` token to iteratively construct sequences.

However, standard discrete diffusion lacks the ability to refine outputs, and its sampling often suffers from a "locked-in" problem: once a token is unmasked, the decision is not reversible, so early prediction biases can cascade into global incoherence. To mitigate this issue, several methods (Ben-Hamu et al., 2025; Campbell et al., 2022; Wang et al., 2025) introduce remasking mechanisms that allow the model to delete and regenerate tokens during inference, providing an "undo" button for refinement. This naturally raises a central question: how should one decide which tokens to remask? Existing training-free samplers typically use model confidence or related reliability scores (Ben-Hamu et al., 2025; Dong et al., 2025; Ouyang et al., 2026; Wang et al., 2025; Zhai et al., 2026) for such a decision. While empirically effective, these methods can generally modify the sampling transition, and in particular, the resulting kernels need not preserve a meaningful target, weakening the probabilistic foundation of diffusion sampling. Another line of work learns remasking or correction policies through additional training (Hersche et al., 2025; Huang et al., 2025; Meshchaninov et al., 2025; Kim et al., 2025; Rout et al., 2025; Schiff et al., 2026; Zhao et al., 2024), but such methods introduce extra training cost, are often task-dependent, and still mostly lack an invariance guarantee. Therefore, existing remasking policies are best viewed as effective heuristic refinement procedures rather than principled samplers.

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

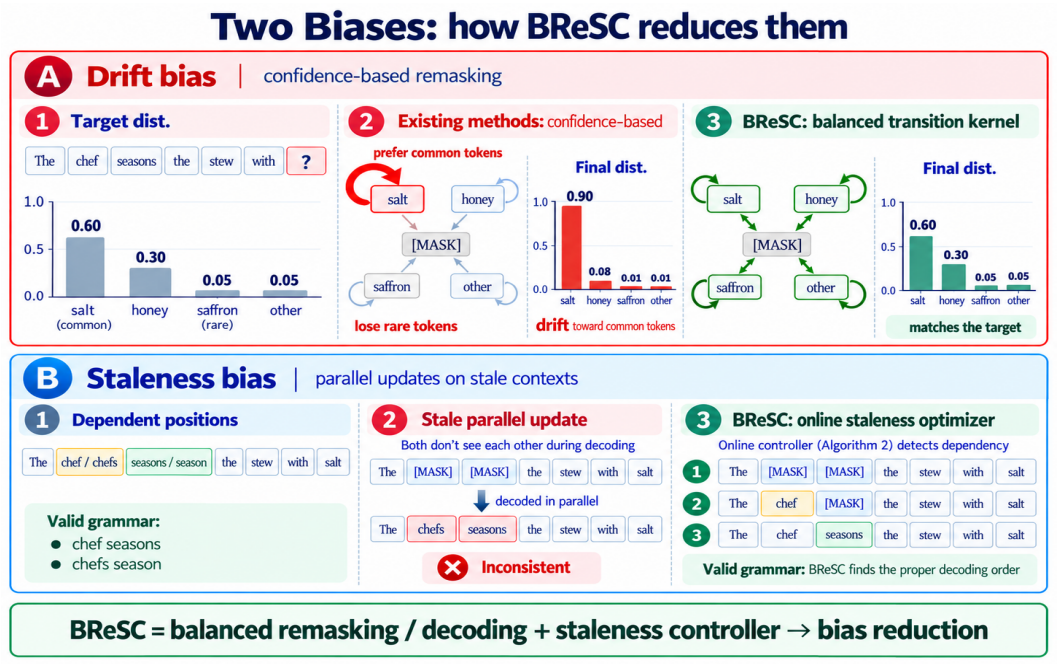

Figure 1: Two layers of remasking bias in the MCMC view. *Drift bias* arises when the local remask/decode kernel does not preserve the chosen target distribution; we address it by balancing the one-site dynamics, making the sequential kernel target-invariant. *Staleness bias* arises in practical parallel sampling, where several tokens are evaluated from one stale snapshot; we address it by optimizing joint updates to reduce estimated interference.

This motivates the following foundational question:

> *Can we design a training-free, parallel remasking sampler which is anchored to a meaningful target distribution?*

To answer this question, we focus on the *remasking correction loop at a fixed noise level* first proposed by Wang et al. (2025) (see Section 2), and formulate such remasking-based inference through the principled lens of **Markov Chain Monte Carlo (MCMC)**. This perspective reveals two fundamental sources of bias that can move the practical parallel sampler away from target-preserving dynamics: (i) a *drift bias* from remasking–decoding transitions that violate detailed balance, a central route to stationarity in MCMC (Levin & Peres, 2017; Robert et al., 2004), and (ii) a *staleness bias* from parallel updates of coupled tokens using stale contexts. We address these two biases separately. First, we introduce an entropy-balanced policy that uses token uncertainty to decide which tokens to remask/decode. We couple these decisions so that the one-site updates satisfy detailed balance and the sequential sweep preserves the chosen target under the ideal-denoiser assumption at a fixed noise level and entropy cache. Second, building on coupling techniques from asynchronous Gibbs sampling (De Sa et al., 2016), we derive a one-sweep discrepancy bound and use it to motivate an online controller that reduces an estimated staleness cost. The controller is implemented through a primal–dual Lagrangian routine, practical at inference time; see Figure 1 for an illustration of the two biases and our solutions.

The resulting algorithm, BReSC (**B**alanced **Re**masking with **S**taleness **C**ontrol), is a fully training-free and plug-and-play parallel sampler for existing MDMs. In the idealized sequential setting, BReSC is free from drift and staleness biases. In practical parallel inference, it uses efficient estimation and adaptation to remain anchored to a target preferred at inference time. Beyond its theoretical grounding, BReSC also improves text-generation efficiency by $\sim 5\times$ in terms of sampling steps and $\sim 2.7\times$ in terms of wall-clock time, and outperforms strong training-free baselines e.g. ReMDM (Wang et al., 2025) on large-scale downstream tasks. Other training-free methods accelerate decoding without remasking (Kim et al., 2026; Wu et al., 2026); our focus is parallel correction of generated tokens.

Contributions. Our contributions are summarized as:

1. We introduce BReSC, a training-free remasking sampler for MDMs. Derived from an MCMC perspective, BReSC anchors one-site laws derived from a target-preserving MCMC,

with a practical adaptive entropy tilt that is effective specifically at inference time, offering a more probabilistically principled sampler.

2. We identify two sources of bias in remasking-based inference: drift bias and staleness bias. Under an ideal denoiser and fixed noise level and entropy cache, the balanced one-site laws yield a target-invariant sequential sweep. We also derive a one-sweep discrepancy bound that motivates online staleness control.
3. We empirically validate BReSC on unconditional text generation and downstream tasks covering arithmetic reasoning, factual QA, code generation, and mathematical reasoning. BReSC achieves state-of-the-art performance in generation quality and sampling efficiency over competitive remasking-based baselines.

2 Remasking as an MCMC and its Sampling Bias

We first present the preliminaries on MDMs and the remasking mechanism. Then we formulate the correction loop phase of remasking, the main object studied in this paper, as MCMC dynamics, following which we identify two errors: drift and staleness.

2.1 Remasking MDM as an MCMC Sampling Problem

We consider length-n sequence generation over a finite vocabulary $\mathcal{V}$. Let p_{data} be the data distribution on $\mathcal{V}^n$, and let $X^\star = (X_1^\star, \dots, X_n^\star) \sim p_{\mathrm{data}}$ denote a clean sequence. Throughout, uppercase letters denote random sequences and lowercase letters denote their realized values. Masked diffusion models augment the vocabulary with a mask symbol $\texttt{[MASK]} \notin \mathcal{V}$ and use the state space

$$\mathcal{X} := (\mathcal{V} \cup \{\texttt{[MASK]}\})^n.$$

For $x = (x_1, \dots, x_n) \in \mathcal{X}$, we write x_{-i} for all coordinates except i. We also write $\texttt{[MASK]}^n := (\texttt{[MASK]}, \dots, \texttt{[MASK]})$ for the all-mask sequence.

Preliminaries on MDMs. We recall the standard MDM noising construction (Sahoo et al., 2024; Shi et al., 2024). For $\alpha \in [0,1]$, where $1-\alpha$ is the masking/noise level, the forward process of an MDM forms a partially masked sequence X^α from $X^\star$ by independently setting

$$X_i^\alpha = \begin{cases} X_i^\star, & \text{with probability } \alpha, \\ \texttt{[MASK]}, & \text{with probability } 1-\alpha. \end{cases}$$

Thus the forward process moves from $X^1 = X^\star$ to the terminal all-mask state $X^0 = \texttt{[MASK]}^n$ as α decreases from 1 to 0. For each fixed α, this process induces the base MDM marginal

$$\Pi_\alpha^{\mathrm{base}}(x) := \mathbb{P}(X^\alpha = x), \qquad x \in \mathcal{X}, \tag{1}$$

where the probability is over both the clean sequence $X^\star \sim p_{\mathrm{data}}$ and the independent masking decisions. In particular, Π_0^{base} puts all mass on $\texttt{[MASK]}^n$, while $\Pi_1^{\mathrm{base}} = p_{\mathrm{data}}$.

Generation reverses this by a backward process: starting from $\texttt{[MASK]}^n$, the sampler increases α and decodes masked coordinates. Let $p_i(\cdot \mid x, \alpha)$ denote the denoiser distribution used to decode coordinate i when $x_i = \texttt{[MASK]}$. Under the standard exact-denoiser idealization, this distribution equals the posterior distribution under the forward masking process:

$$p_i(v \mid x, \alpha) = \mathbb{P}\big(X_i^\star = v \mid X_i^\alpha = \texttt{[MASK]},\, X_{-i}^\alpha = x_{-i}\big), \qquad x_i = \texttt{[MASK]},\ v \in \mathcal{V}. \tag{2}$$

In practice, a well-learned MDM denoiser approximates this posterior.

The family $\{\Pi_\alpha^{\mathrm{base}}\}_{\alpha\in[0,1]}$ is the base MDM marginal path from the all-mask point mass to the clean data distribution. Standard MDM inference aims to follow this path in reverse. However, practical MDM samplers often decode multiple coordinates in parallel from the same partially masked state, so their distribution can deviate from the intended marginal at the corresponding noise level (Chen et al., 2025). This motivates the fixed-α remasking loop studied below, where masked coordinates may be decoded and revealed coordinates may be masked again.

Remasking MDM as MCMC Sampling. Standard reverse MDM decoding is one-way: once a coordinate is revealed, a decode-only sampler keeps it fixed subsequently. With parallel decoding, multiple coordinates are predicted from the same partially masked input and a coordinate decoded incorrectly can remain in the sample and influence later predictions, creating the "locked-in" problem.

At a fixed α, these effects can make the distribution of the sampler's current state differ from the corresponding fixed-noise marginal Π_α^{base}.

Remasking methods address this issue by allowing revealed coordinates to be revised; ReMDM (Wang et al., 2025) is a representative example, which inserts an extra correction loop at a fixed α. A ReMDM-style sampler can be viewed as three stages: (a) a standard reverse-MDM stage that runs from the initial $\texttt{[MASK]}^n$ to a prescribed noise level $1-\alpha$; (b) a fixed-α correction loop, in which revealed coordinates may be remasked and masked coordinates may be decoded; and (c) a final standard reverse-MDM stage that resumes the denoising schedule towards $\alpha = 1$. We **focus only on stage (b), the fixed-α correction loop**; (a) and (c) are kept as in the base MDM sampler.

During stage (b), α is fixed, so each remask–decode update is a transition on $\mathcal{X}$, and repeated updates define a *Markov chain*. The MCMC viewpoint is to treat this loop as an MC sampler whose transition probabilities are designed relative to an explicit target distribution. The natural baseline target is Π_α^{base}; Section 3.1 later introduces the entropy-tilted target used by BReSC. We next define the coordinate-wise update laws and the full-sequence kernels they induce.

Remasking Policy and Transition Kernel for x_i. At each coordinate, the two nontrivial moves are decoding a mask and remasking a revealed token. We define *remasking and decoding policies*

$$r_i : \{x \in \mathcal{X} : x_i \neq \texttt{[MASK]}\} \to [0,1], \qquad d_i : \{x \in \mathcal{X} : x_i = \texttt{[MASK]}\} \to [0,1], \tag{3}$$

where $r_i(x)$ is the probability of remasking coordinate i when $x_i \neq \texttt{[MASK]}$, and $d_i(x)$ is the probability of decoding coordinate i when $x_i = \texttt{[MASK]}$. Given x, these policies induce the following law for updating x_i:

$$\ell_i(\cdot \mid x) = \begin{cases} (1 - r_i(x))\delta_{x_i} + r_i(x)\delta_{\texttt{[MASK]}}, & x_i \neq \texttt{[MASK]}, \\ (1 - d_i(x))\delta_{\texttt{[MASK]}} + d_i(x)p_i(\cdot \mid x, \alpha), & x_i = \texttt{[MASK]}, \end{cases} \tag{4}$$

where δ_z denotes a point mass at z. Thus $\ell_i(\cdot \mid x)$ is the distribution of the new value of coordinate i after one local remasking–decoding update. With ℓ_i, we define the corresponding *Markov kernel* for coordinate i, on the full sequence space with all other coordinates fixed:

$$L_i(x, y) = \ell_i(y_i \mid x)\, \delta_{x_{-i}}(y_{-i}). \tag{5}$$

Now L_i defines the transition of the single coordinate i, and a global transition for all coordinates requires a composition of all L_i's; see below.

Transition Kernel for x. In an MCMC setting, probably the simplest Markov kernel built from these one-site updates is the *fresh sequential sweep kernel*,

$$P := L_n \circ L_{n-1} \circ \cdots \circ L_1. \tag{6}$$

where $\circ$ denotes Markov-kernel composition: starting from x, we apply L_1 first, then L_2, and so on until L_n. This sequential kernel P is the clean MCMC reference against which we study whether invariance can be achieved.

In practice, however, MDM inference is parallel. Consequently, the kernel is no longer sequentially sweeping through L_i as in (6), but instead

$$\widetilde{P}(x, y) := \bigotimes_{i=1}^n \ell_i(y_i \mid x), \tag{7}$$

which we refer to as a *cached parallel sweep*. Here $\bigotimes_{i=1}^n \ell_i$ denotes the product distribution over ℓ_i's. Equivalently, conditional on x, the updated coordinates Y_i are sampled independently with $Y_i \sim \ell_i(\cdot \mid x)$. P and $\widetilde{P}$ use the same one-site laws ℓ_i's, but execute them differently: P refreshes the context after each coordinate update, whereas $\widetilde{P}$ evaluates all coordinates from the same context.

With the formulation above, the rest of the paper studies two questions. First, how should we choose the remasking–decoding policies r and d so that the sequential sweep kernel P preserves a prespecified target distribution? Second, after the target is preserved, how large is the discrepancy between the practical parallel $\widetilde{P}$ and the sequential P, and how can it be controlled during inference?

2.2 Bias I: Drift Bias from Unbalanced Kernels

The remasking policy (3) turns the original denoising distribution p_i into a new one-site kernel L_i through the rates r_i and d_i. For the fixed-α remasking loop, we ask when this kernel preserves a

Algorithm 1 BReSC: Balanced Remasking with Staleness Control

Require: pre-trained denoiser p, length n, timesteps T, time grid $\{\tau_t = (T-t)/T\}$, loop interval $(t_{\text{off}}, t_{\text{on}}]$, controller flag ctrl.

Initialize $x_0 = (\texttt{[MASK]}, \ldots, \texttt{[MASK]})$, entropy cache h, and controller state $\mathcal{S}$.
for $t = 1, \ldots, T$ **do**
  Run the denoiser at x_{t-1}, refresh h_i for currently masked sites and hold it fixed during this loop step.
  if $\tau_t \notin (t_{\text{off}}, t_{\text{on}}]$ **then**
    Sample x_t from the standard MDM laws (standard decoding).
  else
    Set the entropy-balanced policy by (13), and form $\ell_{t,i}(\cdot) := \ell_i(\cdot \mid x_{t-1})$ using (4).
    if ctrl **then**
      $x_t, \mathcal{S} \leftarrow \textsc{OnlineStale}(x_{t-1}, \{\ell_{t,i}\}_{i=1}^n, \mathcal{S})$.
    else
      Sample $x_{t,i} \sim \ell_{t,i}(\cdot)$ independently for all i.
return x_T.

desired MCMC target. We first ask when the one-site kernel L_i is consistent with a generic target distribution Π defined on $\mathcal{X}$. For $x \in \mathcal{X}$, we write $x_i^{\leftarrow a}$ for the sequence obtained by replacing x_i with $a \in \mathcal{V} \cup \{\texttt{[MASK]}\}$. Its one-site conditional is

$$\Pi_i(a \mid x_{-i}) := \frac{\Pi(x_i^{\leftarrow a})}{\sum_{b \in \mathcal{V} \cup \{\texttt{[MASK]}\}} \Pi(x_i^{\leftarrow b})}, \qquad a \in \mathcal{V} \cup \{\texttt{[MASK]}\}. \tag{8}$$

A central route to stationarity in the literature of MCMC is **detailed balance** (Levin & Peres, 2017; Robert et al., 2004). Specifically in our setting, L_i preserves Π if it satisfies the following *coordinate-wise detailed-balance equation*:

$$\Pi(x) L_i(x, y) = \Pi(y) L_i(y, x), \qquad x, y \in \mathcal{X}. \tag{9}$$

For the one-site law ℓ_i in (4), the only nontrivial moves are between [MASK] and a revealed token. Thus, by setting $x = (x_1, \ldots, v, \ldots, x_n)$, $y = (x_1, \ldots, \texttt{[MASK]}, \ldots, x_n)$ in (9) and plugging in (8), we obtain for any x_{-i} and $v \in \mathcal{V}$,

$$\Pi_i(v \mid x_{-i})\, r_i(x_i^{\leftarrow v}) = \Pi_i(\texttt{[MASK]} \mid x_{-i})\, d_i(x)\, p_i(v \mid x, \alpha). \tag{10}$$

(10) shows that remasking and decoding cannot be chosen independently: to preserve a target, the remasking rate r_i at the revealed neighbor $x_i^{\leftarrow v}$ must be coupled with the decoding rate d_i at the state x. If (10) is violated, the resulting local kernel may preserve a distribution different from the intended target. We refer to this mismatch as **drift bias**.

2.3 Bias II: Staleness Bias from Cached Parallel Sweeps

If each L_i satisfies coordinate-wise detailed balance (9) with respect to Π, then the fresh sequential sweep P (6) preserves Π:

Proposition 1 (Fresh-sweep invariance)**.** *If each L_i satisfies* (9) *with respect to* Π*, then the fresh sweep P in* (6) *is* Π*-invariant. In addition, if P is irreducible and aperiodic on $\mathcal{X}$, then* Π *is the unique stationary distribution on* $\mathcal{X}$*.*

Under the additional irreducibility and aperiodicity conditions in Proposition 1, repeated fresh sweeps converge to Π from any initial state. The proof is deferred to Appendix B.1.

Practical MDM sampling, however, does not run P. It uses $\widetilde{P}$ in (7), where all coordinates are updated in parallel using laws evaluated at the same stale x. Therefore, even when every L_i is balanced, the cached sweep $\widetilde{P}$ generally differs from the fresh sweep P, because the one-site laws in $\widetilde{P}$ do not see changes made by other coordinates. Following the terminology of *asynchronous MCMC* (De Sa et al., 2016), we call this systematic discrepancy **staleness bias**.

3 Balancing and Staleness Control via BReSC

The previous section separates remasking bias into drift and parallel staleness biases. We now introduce BReSC, which accordingly has two components: a *balancing layer* chooses the remasking–decoding policies r, d, and a *staleness-control layer* changes joint coupling to suppress co-updates with larger staleness. The practical BReSC algorithm is grounded in MCMC principles, with effective estimation and adaptation specifically for inference-time performance.

3.1 TARGET DISTRIBUTIONS FOR BALANCED REMASKING

This section asks which target to sample from at **inference time**. Section 2.1 introduced the base target Π_α^{base} in (1). This target is the natural MDM marginal at noise level $1-\alpha$. At inference time, modifying a model's sampling distribution can improve generation quality, as shown by nucleus sampling, locally typical sampling, and MCMC sampling (Holtzman et al., 2020; Meister et al., 2023; Karan & Du, 2026). Confidence-based masked decoding similarly uses token reliability to guide generation (Chang et al., 2022; Ben-Hamu et al., 2025). We encode an uncertainty-based preference in an explicit *entropy-tilted target* and derive target-preserving reference updates.

During sampling, let $h = (h_1, \ldots, h_n)$ be an entropy cache, where h_i is the Shannon entropy of the token marginal distribution p_i conditional on the previous context at coordinate i. For $\beta \geq 0$, define the entropy-tilted target

$$\Pi_{\alpha,h}^{\text{ent}}(x) \propto \Pi_\alpha^{\text{base}}(x) \exp\big(-\beta \textstyle\sum_{i:\, x_i \neq \texttt{[MASK]}} h_i\big). \tag{11}$$

Here β controls the strength of the entropy tilt and serves as an *inverse-temperature* parameter, while h is treated as an external cached vector within the fixed-noise loop. With h fixed, the tilt favors revealing low-uncertainty coordinates and leaving uncertain ones masked. For a fixed mask pattern, token values retain their relative probabilities under the base target. This motivates refining uncertain positions, but target preservation alone does not guarantee better text. When $\beta = 0$, it reduces to the base target Π_α^{base}. Given this target, the next subsection derives remasking and decoding rates whose sequential kernel P is target-preserving.

3.2 BALANCING THE STATE-DEPENDENT POLICIES

For a fixed target Π, Section 2.2 shows that removing drift bias amounts to satisfying the detailed-balance condition (10) with proper r_i and d_i. The theorem below gives balanced policies for the base target Π_α^{base} as a reference case and for the entropy-tilted target $\Pi_{\alpha,h}^{\text{ent}}$ used by BReSC.

Theorem 1 (Balanced remasking policies)**.** *Assume the ideal-denoiser condition* (2) *and fix* $\alpha \in (0,1)$. *Let* $\kappa_\alpha := \alpha/(1-\alpha)$. *For the base target* Π_α^{base}*, for any* $b \in [0, \min\{1, \kappa_\alpha^{-1}\}]$*, the policy*

$$r_i(x_i^{\leftarrow v}) = b, \qquad d_i(x) = \kappa_\alpha b \tag{12}$$

is well-defined and satisfies the coordinate-wise detailed-balance condition (10)*.*

For the entropy-tilted target $\Pi_{\alpha,h}^{\text{ent}}$ *with fixed entropy cache* h*, let* $\beta_R, \beta_D \geq 0$ *satisfy* $\beta_R + \beta_D = \beta$ *for some* $\beta \geq 0$*. Then, for any* $b \in \big[0,\ \min_{1\leq i\leq n} \min\big\{e^{-\beta_R h_i},\ \kappa_\alpha^{-1} e^{\beta_D h_i}\big\}\big]$*, the policy*

$$r_i(x_i^{\leftarrow v}) = b e^{\beta_R h_i}, \qquad d_i(x) = \kappa_\alpha b e^{-\beta_D h_i} \tag{13}$$

is well-defined and satisfies (10)*.*

The proof is in Appendix B.2. The theorem gives the balancing layer of BReSC; by Proposition 1, the corresponding sequential sweep P is target-invariant. For the base target Π_α^{base}, the mask–token odds are controlled only by α, so constant r_i and d_i suffice. For the entropy-tilted target $\Pi_{\alpha,h}^{\text{ent}}$, the cached entropy h_i modifies the two rates so that high-entropy revealed coordinates are more likely to be remasked, while high-entropy masked coordinates are less likely to be decoded.

This also clarifies why *confidence-guided remasking*, as used in ReMDM (Wang et al., 2025) and (Ouyang et al., 2026; Wang et al., 2025; Zhai et al., 2026), is generally not balanced. For fixed x, the balance equation (10) must hold for every token v with the same decoding rate $d_i(x)$. Confidence-based remasking rates generally vary with v, so this consistency need not hold. Consequently, confidence-based policy does not automatically induce a P whose target is explicitly defined, and can suffer from unwanted drift bias. By contrast, an **entropy-balanced policy** (13) avoids this issue because the cached entropy h_i is a coordinate-level score and is independent of v.

Algorithm 1 summarizes the practical implementation of our entropy-balanced policy within a general MDM, with the flag `ctrl` enabling the online staleness controller introduced next. The sampler is plug-and-play: it requires no retraining and leaves the denoiser distribution $p_i(\cdot \mid x, \alpha)$ unchanged.

Inference-time adaptation. Practical inference-time decoding can improve generation quality by modifying a pretrained model's sampling distribution (Holtzman et al., 2020; Meister et al., 2023; Karan & Du, 2026). Motivated by this, BReSC uses an entropy-tilted target and refreshes the entropy cache as generation proceeds. Theorem 1 is established for a fixed h, while refreshing h in practice is a theory-guided adaptation for better capturing the evolving inference-time preferred target, as supported by our experiments in Section 4.

3.3 Staleness Bias and Online Control

The balanced construction above removes drift bias for the sequential sweep kernel P. However, sequential sampling is too slow for large models, and practical MDM inference uses the parallel sweep $\widetilde{P}$ in (7). We now bound the resulting one-sweep staleness discrepancy between P and $\widetilde{P}$ and use this bound to motivate an online co-update proxy for controlled parallel execution.

3.3.1 Influence View of Cached-Sweep Staleness Bias

From an MCMC viewpoint, staleness has been studied in asynchronous Gibbs sampling and chromatic Gibbs sampling (De Sa et al., 2016; Terenin et al., 2020; Gonzalez et al., 2011), but existing bounds are often worst-case and difficult to estimate. Here, we exploit the fact that staleness bias arises when interacting coordinates are updated together.

Past studies suggest that a cached parallel MCMC sweep is close to a fresh sequential sweep when a changed coordinate has little effect on the update law of the other simultaneously changed coordinates. This motivates the following definition of **expected marginal influence**: for coordinates $i \neq j$, define

$$A_{ij}(x) := \mathbb{E}_{V_j \sim \ell_j(\cdot|x)}\Big[\,\mathrm{TV}\left(\ell_i(\cdot \mid x), \ell_i(\cdot \mid x_j^{\leftarrow V_j})\right)\Big]. \tag{14}$$

The quantity $A_{ij}(x)$ measures the expected change in the update law at i caused by first updating j, automatically discounting rare or low-impact token changes. It therefore captures the local co-activation cost of updating i and j in the same parallel sweep.

Theorem 2 (Bound on staleness bias)**.** *Let $X^{(0)} = x$, and let $X^{(k)}$ denote the state after the first k coordinates have been updated by the sequential sweep P. Then the discrepancy bound between P as in* (6) *and $\widetilde{P}$ as in* (7) *is as follows:*

$$\mathrm{TV}\left(\widetilde{P}(x,\cdot), P(x,\cdot)\right) \leq \textstyle\sum_{1\leq j<i\leq n} \mathbb{E}\Big[A_{ij}\big(X^{(j-1)}\big)\Big]. \tag{15}$$

The proof is given in Appendix B.3. Theorem 2 shows that staleness is costly when strongly dependent coordinates are updated in the same cached sweep. Unlike classical asynchronous Gibbs bounds based on global worst-case delay or coupling constants (De Sa et al., 2016; Terenin et al., 2020), our bound deals with a one-sweep discrepancy between $\widetilde{P}$ and P and more crucially, is **estimable**.

Online Proxy of A_{ij}. Evaluating $A_{ij}(x)$ exactly requires extra model passes at counterfactual inputs $x_j^{\leftarrow v}$, making it too expensive during inference. We therefore estimate a tractable online proxy of the influence matrix $A(x) := (A_{ij}(x))_{i,j=1}^n$ without additional counterfactual denoiser passes. The key idea is that the observed influence can be organized as a *Boolean function* of which coordinates change: each coordinate either changes or stays fixed, and its Fourier expansion (O'Donnell, 2014) provides a natural first-order pairwise proxy for $A_{ij}(x)$. Appendix C.1 gives the full derivation and the higher-order residual. During inference, we maintain an exponential-moving-average estimate $\widehat{A}_t^{(1)}$ of this proxy and form W_t by symmetrizing and normalizing it. W_t is the only object passed to the controller below.

3.3.2 Online Optimization for Staleness Control

Our staleness bound suggests that, at each diffusion step, we should avoid changing highly coupled coordinates simultaneously; see Figure 1. This can be formulated as a decision-making problem over which tokens are changed in the current parallel sweep. Let $C \in \{0,1\}^n$ denote the binary change pattern, where $C_i = 1$ means that coordinate i changes. A deterministic choice of C is often too restrictive, so we instead choose a distribution μ over all 2^n possible change patterns.

The key constraint is that this joint choice must preserve the one-coordinate transition marginals of the balanced law. Let $\sigma_{t,i} := 1 - \ell_i(x_{t,i} \mid x_t)$ be the marginal probability that coordinate i changes under $\ell_i(\cdot \mid x_t)$. We require $\mathbb{E}_{C\sim\mu}[C_i] = \sigma_{t,i}$ for all i. When $C_i = 1$, we sample the new value from the move-only law proportional to $\ell_i(a \mid x_t)\mathbf{1}\{a \neq x_{t,i}\}$.

With W_t defined in the last subsection, for a change pattern $c \in \{0,1\}^n$, we define its staleness cost as $s_t(c) := \sum_{1\leq i<j\leq n} W_{t,ij} c_i c_j$. This cost penalizes simultaneous changes of highly coupled coordinates. We choose μ by solving the following **KL-regularized online optimization** problem

$$\min_{\mu\in\Delta(\{0,1\}^n)} \quad \mathbb{E}_{C\sim\mu}\big[s_t(C)\big] + \lambda^{-1} D_{\mathrm{KL}}(\mu\|\nu_t), \qquad \text{s.t. } \mathbb{E}_{C\sim\mu}[C_i] = \sigma_{t,i},\ i = 1,\dots,n, \tag{16}$$

Algorithm 2 ONLINESTALE: Online Staleness Controller

Require: Current state x; balanced one-site laws $\{\ell_i(\cdot \mid x)\}_{i=1}^n$; controller state $\mathcal{S} = (\widehat{A}^{(1)}, \theta, \lambda)$.
1: Set $\sigma_i \leftarrow 1 - \ell_i(x_i \mid x)$, for $i = 1, \dots, n$.
2: For each $\sigma_i > 0$, define $q_i(a) := \ell_i(a \mid x)\mathbf{1}\{a \neq x_i\}/\sigma_i,\ a \in \mathcal{V} \cup \{\texttt{[MASK]}\}$; coordinates with $\sigma_i = 0$ are fixed.
3: Solve the online optimization (16) using the warm-started dual variable θ.
4: Sample a binary change pattern $C \sim \mu_\theta$.
5: Set $y_i = x_i$ when $C_i = 0$, and $y_i = Z_i$ with $Z_i \sim q_i(\cdot)$ when $C_i = 1$.
6: Store the updated θ in $\mathcal{S}$.
7: **return** $y, \mathcal{S}$.

where $\lambda > 0$ and $\nu_t(c) := \prod_{i=1}^n \sigma_{t,i}^{c_i}(1 - \sigma_{t,i})^{1-c_i}$ is the independent baseline law induced by the balanced policy in Section 3.2. The objective favors low-staleness joint update patterns, while the constraints preserve the coordinate-wise balanced transition law.

Primal–dual solution. Optimizing (16) directly over the 2^n-dimensional distribution μ is too expensive during inference. Instead, we note that its Lagrangian dual offers a cleaner solution. With a dual variable $\theta \in \mathbb{R}^n$ for the marginal constraints, since (16) is convex, strong duality (Boyd & Vandenberghe, 2004) ensures that (16) is equivalent to solving $\max_{\theta \in \mathbb{R}^n} \min_{\mu \in \Delta(\{0,1\}^n)} \mathcal{L}(\mu, \theta)$, where $\mathcal{L}(\mu, \theta) = D_{\mathrm{KL}}(\mu \| \nu_t) + \lambda \mathbb{E}_\mu[s_t(C)] - \theta^\top \big(\mathbb{E}_\mu[C] - \sigma_t\big)$ is the Lagrangian. The inner minimization over μ has the closed-form solution

$$\mu_\theta(c) = \frac{\nu_t(c) \exp\{\theta^\top c - \lambda s_t(c)\}}{Z_t(\theta)}, \qquad Z_t(\theta) = \textstyle\sum_{c \in \{0,1\}^n} \nu_t(c) \exp\{\theta^\top c - \lambda s_t(c)\}.$$

When $s_t(c) = \sum_{i<j} W_{t,ij} c_i c_j$, this is a binary Gibbs law: large $W_{t,ij}$ penalizes co-activating influential pairs, while θ enforces the desired one-site move probabilities. Plugging the inner solution back gives the n-dimensional dual problem

$$\min_{\theta \in \mathbb{R}^n} F_t(\theta) := \log Z_t(\theta) - \theta^\top \sigma_t.$$

We solve this by a subsampled Newton method, with θ warm-started from the previous diffusion step; see Appendix C.4 for details. Staleness control is summarized in ONLINESTALE (Algorithm 2).

4 EXPERIMENTS

We evaluate BReSC on unconditional text generation and downstream language-modeling benchmarks. Our goal is to test whether the theoretically motivated balance and staleness-control mechanisms translate into practical gains for MDM inference.

4.1 UNCONDITIONAL TEXT GENERATION

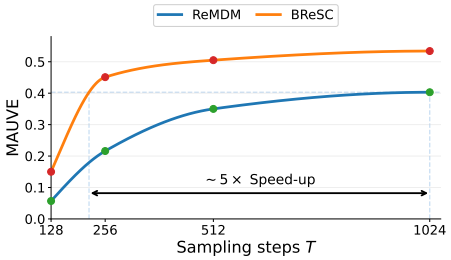

Figure 2: Interpolation indicates that BReSC reaches the ReMDM $T = 1024$ MAUVE level with approximately $5\times$ fewer steps.

We first evaluate unconditional text generation using the pretrained MDM checkpoint of Sahoo et al. (2024), trained on OpenWebText (OWT) (Gokaslan et al., 2019) with the GPT-2 tokenizer (Radford et al., 2019). Following Pillutla et al. (2021); Wang et al. (2025), we generate 5000 independent samples of length $n = 1024$ and vary the number of sampling steps $T \in \{128, 256, 512, 1024\}$. We report MAUVE (Pillutla et al., 2021), generative perplexity (PPL) computed by a GPT-2-large oracle (Radford et al., 2019), and entropy.

As noted in prior work (Zheng et al., 2024; Wang et al., 2025), PPL alone can be misleading for MDMs, since low perplexity may coexist with reduced diversity or repetitive generations. We therefore use MAUVE as the primary metric, as it measures distributional similarity while balancing quality and diversity (Pillutla et al., 2021; Wang et al., 2025). We follow the public ReMDM decoding, remasking-loop, and evaluation protocol (Wang et al., 2025), replacing only the sampler with BReSC; Appendix D.1 gives the settings.

Results. Table 1 shows that BReSC substantially improves both generation quality and efficiency. At an equal budget, BReSC improves MAUVE over ReMDM by $2.63\times$ and $2.09\times$ for $T = 128, 256$,

respectively. BReSC at $T = 256$ achieves MAUVE 0.451, exceeding ReMDM's 0.403 at $T = 1024$ with **one quarter as many sampling steps**; see Figure 2. Against vanilla MDM, the MAUVE improvement ranges from $10.0\times$ to $19.6\times$ across all T. Notably, BReSC also outperforms a recent training-based baseline PRISM (Kim et al., 2025) on MAUVE for every T, despite requiring no additional adapter training. Finally, BReSC consistently improves over BReSC without staleness control in both MAUVE and PPL, confirming the practical value of Algorithm 2. Appendix E.1.4 also applies Algorithm 2 to ReMDM, showing that staleness control alone can improve inference. We also note that Algorithm 2 adds the computational overhead of solving (16), where BReSC takes $1.84\times$ the ReMDM time per sample; see Table 7, Appendix E.1.2. However, combined with the estimated $\sim 5\times$ reduction in sampling steps, this still gives an estimated $\mathbf{2.7\times}$ **wall-clock speedup** at matched MAUVE. This estimated net gain supports BReSC's practical value despite its overhead; optimizing the solver for speed remains future work. To assess the benefit of our balanced design (10) beyond entropy-based remasking alone, we compare with an *unbalanced variant of BReSC* that retains the same entropy-based remasking but breaks (10). The gains in MAUVE and PPL support the balanced design; see Appendix E.1.1.

Table 1: Unconditional text generation on OpenWebText. We report MAUVE (higher is better), generative perplexity (PPL, lower is better), and entropy as a diversity diagnostic across different sampling steps T. The primary comparison is among training-free samplers; PRISM† is included as a training-based reference. Best results are in **bold**; second best are underlined.

	MAUVE ↑				PPL ↓				Entropy			
Method / $T =$	128	256	512	1024	128	256	512	1024	128	256	512	1024
MDM	0.015	0.023	0.031	0.042	61.5	55.8	53.0	51.3	5.52	5.49	5.48	5.46
ReMDM	0.057	0.216	0.350	0.403	42.5	30.5	21.1	28.6	5.43	5.34	5.21	5.38
BReSC (no SC)	0.119	0.368	0.490	0.496	29.2	16.8	15.3	12.0	5.40	5.32	5.24	5.13
BReSC	**0.150**	**0.451**	**0.505**	**0.534**	**28.9**	**16.5**	**15.1**	**11.6**	5.39	5.32	5.22	5.11
PRISM†	0.118	0.294	0.423	0.527	21.5	18.0	16.4	15.3	5.18	5.15	5.12	5.10

†PRISM is training-based and requires additional adapter training; all other methods are training-free.

4.2 Downstream Benchmark with LLMs

We evaluate downstream performance using LLaDA-8B-Instruct (Nie et al., 2025) with BReSC as its remasking sampler. We benchmark on Countdown (Gandhi et al., 2024; Ye et al., 2024), TruthfulQA (Lin et al., 2022), HumanEval (Chen et al., 2021), and GSM8K (Cobbe et al., 2021), covering arithmetic planning, factual question answering, code generation, and mathematical reasoning. We report pass@1 for Countdown, HumanEval, and GSM8K (under both flexible and restricted matching protocols). For TruthfulQA, we report ROUGE scores (Lin, 2004). We follow the setup of ReMDM (Wang et al., 2025) for Countdown and TruthfulQA with generation length $n = 32$; and Schiff et al. (2026) for HumanEval and GSM8K, using generation length $n = 256$ with block diffusion of size 32; see Appendices D.2 and E.2.

Results. Table 2 shows that BReSC consistently improves downstream performance over vanilla LLaDA and LLaDA+ReMDM. Compared with ReMDM, BReSC improves Countdown by 5.3 points and HumanEval by 7.3 points. On GSM8K, the gains are 1.7 points under flexible extraction and 8.3 under restricted extraction.

Table 2: Downstream performance of LLaDA-8B-Instruct with different remasking samplers. We report pass@1 for Countdown, HumanEval, and GSM8K, and ΔROUGE-1/2/L for TruthfulQA. For GSM8K, results are reported as flexible / restricted.

Method	**Countdown**	**TruthfulQA**	**HumanEval**	**GSM8K**
LLaDA	45.5	26.9 / 29.8/ 27.0	39.6	77.3 / 39.1
LLaDA+ReMDM	46.0	28.7/ 31.2 / 28.8	39.0	78.7 / 40.3
LLaDA+BReSC	**51.3**	**29.1/ 31.6/ 29.2**	**46.3**	**80.4 / 48.6**

5 Conclusion

We introduced BReSC, a training-free and plug-and-play parallel sampler for MDM inference. The central message of this work is that remasking should not be viewed only as a heuristic editing mechanism, but as a sampling procedure from an MCMC viewpoint. From this perspective, we identified drift bias and staleness bias as two key obstacles in existing remasking samplers, and showed that BReSC provides control over both of them. BReSC improves both unconditional generation and downstream LLM benchmarks, demonstrating that principled sampling design can yield practical gains. These results suggest that classical principles from MCMC and optimization can provide a useful foundation for the next generation of discrete diffusion inference methods.

AI USE STATEMENT

We used generative AI tools to assist with portions of the code, generate some figures, check mathematical proofs, and improve the clarity and style of the manuscript. The authors ran, tested, and manually inspected the AI-assisted code, carefully reviewed the figures and compared them with the underlying results or intended content, and independently verified the mathematical arguments. The authors reviewed the AI-assisted work and take responsibility for the final content of this paper, including its text, claims, code, and figures.

ETHICS STATEMENT

BReSC modifies inference for pretrained language models and can inherit their biases, factual errors, and potential to generate harmful content. Improved generation quality or efficiency does not establish the safety or reliability of generated text; applications require appropriate evaluation and safeguards.

REPRODUCIBILITY STATEMENT

Algorithms 1 and 2 specify the sampler and controller. Appendix B gives the proofs, while Appendix C provides the influence estimator and controller implementation details. Appendix D documents the model checkpoints, datasets, sampling settings, and evaluation protocols; the additional results include ablations and controller timing measurements.

REFERENCES

Pierre Alquier, Nial Friel, Richard Everitt, and Aidan Boland. Noisy monte carlo: Convergence of markov chains with approximate transition kernels. *Statistics and Computing*, 26(1):29–47, 2016.

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

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.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*, 2025.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Dake Bu, Wei Huang, Andi Han, Hau-San Wong, Qingfu Zhang, Taiji Suzuki, and Atsushi Nitanda. Dprm: A plug-in doob h transform-induced token-ordering module for diffusion language models. *arXiv preprint arXiv:2604.24357*, 2026.

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.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Sitan Chen, Kevin Cong, and Jerry Li. Optimal inference schedules for masked diffusion models. *arXiv preprint arXiv:2511.04647*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

Christopher De Sa, Chris Ré, and Kunle Olukotun. Ensuring rapid mixing and low bias for asynchronous gibbs sampling. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1567–1576. PMLR, 2016.

Yihong Dong, Zhaoyu Ma, Xue Jiang, Zhiyuan Fan, Jiaru Qian, Yongmin Li, Jianha Xiao, Zhi Jin, Rongyu Cao, Binhua Li, et al. Saber: An efficient sampling with adaptive acceleration and backtracking enhanced remasking for diffusion language model. *arXiv preprint arXiv:2510.18165*, 2025.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus, 2019.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel gibbs sampling: From colored fields to thin junction trees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 324–332. JMLR Workshop and Conference Proceedings, 2011.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. *arXiv preprint arXiv:2510.17206*, 2025. doi: 10.48550/arXiv.2510.17206. Accepted at ICLR 2026.

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective dllms. *arXiv preprint arXiv:2507.18578*, 2025.

Zemin Huang, Yuhang Wang, Zhiyang Chen, and Guo-Jun Qi. Don't settle too early: Self-reflective remasking for diffusion language models. *arXiv preprint arXiv:2509.23653*, 2025.

Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. In *International Conference on Learning Representations*, volume 2026, pp. 476–494, 2026.

Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. Fine-tuning masked diffusion for provable self-correction. *arXiv preprint arXiv:2510.01384*, 2025.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models. *Advances in Neural Information Processing Systems*, 38:92267–92301, 2026.

Sanghyun Lee, Sunwoo Kim, Seungryong Kim, Jongho Park, and Dongmin Park. Effective test-time scaling of discrete diffusion through iterative refinement. *arXiv preprint arXiv:2511.05562*, 2025.

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

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

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 2023.

Viacheslav Meshchaninov, Egor Shibaev, Artem Makoian, Ivan Klimov, Danil Sheshenya, Andrei Malinin, Nikita Balagansky, Daniil Gavrilov, Aibek Alanov, and Dmitry Vetrov. Guided star-shaped masked diffusion. *arXiv preprint arXiv:2510.08369*, 2025. doi: 10.48550/arXiv.2510.08369.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

Yidong Ouyang, Panwen Hu, Zhengyan Wan, Zhe Wang, Liyan Xie, Dmitriy Bespalov, Ying Nian Wu, Guang Cheng, Hongyuan Zha, and Qiang Sun. Training-free self-correction for multimodal masked diffusion models. *arXiv preprint arXiv:2602.02927*, 2026.

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Anru R. Zhang, Michael Bronstein, Alexander Tong, and Avishek Joey Bose. Planner aware path learning in diffusion language models training. *arXiv preprint arXiv:2509.23405*, 2026. doi: 10.48550/arXiv.2509.23405.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828, 2021.

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.

Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 2004.

Litu Rout, Constantine Caramanis, and Sanjay Shakkottai. Anchored diffusion language model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL `https://openreview.net/forum?id=E8adS5srds`.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Shreshth Saini, Avinab Saha, Balu Adsumilli, Neil Birkbeck, Yilin Wang, and Alan C. Bovik. Tabes: Trajectory-aware backward-on-entropy steering for masked diffusion models. *arXiv preprint arXiv:2602.00250*, 2026. doi: 10.48550/arXiv.2602.00250.

Yair Schiff, Omer Belhasin, Roy Uziel, Guanghan Wang, Marianne Arriola, Gilad Turok, Michael Elad, and Volodymyr Kuleshov. Learn from your mistakes: Self-correcting masked diffusion models. *arXiv preprint arXiv:2602.11590*, 2026.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

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

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

Alexander Terenin, Daniel Simpson, and David Draper. Asynchronous gibbs sampling. In *International Conference on Artificial Intelligence and Statistics*, pp. 144–154. PMLR, 2020.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. *arXiv preprint arXiv:2503.00307*, 2025.

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. In *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026.

Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. Energy-based diffusion language models for text generation. *arXiv preprint arXiv:2410.21357*, 2024.

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.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Kevin Zhai, Sabbir Mollah, Zhenyi Wang, and Mubarak Shah. Core: Context-robust remasking for diffusion language models. *arXiv preprint arXiv:2602.04096*, 2026.

Yixiu Zhao, Jiaxin Shi, Feng Chen, Shaul Druckmann, Lester Mackey, and Scott Linderman. Informed correctors for discrete diffusion models. *arXiv preprint arXiv:2407.21243*, 2024.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.

CONTENTS

1 Introduction **1**

2 Remasking as an MCMC and its Sampling Bias **3**

2.1 Remasking MDM as an MCMC Sampling Problem 3

2.2 Bias I: Drift Bias from Unbalanced Kernels 4

2.3 Bias II: Staleness Bias from Cached Parallel Sweeps 5

3 Balancing and Staleness Control via BReSC **5**

3.1 Target Distributions for Balanced Remasking 6

3.2 Balancing the State-Dependent Policies 6

3.3 Staleness Bias and Online Control 7

3.3.1 Influence View of Cached-Sweep Staleness Bias 7

3.3.2 Online Optimization for Staleness Control 7

4 Experiments **8**

4.1 Unconditional Text Generation 8

4.2 Downstream Benchmark with LLMs 9

5 Conclusion **9**

A Related Work **15**

B Proofs and Additional Derivations **16**

B.1 Proof of Proposition 1 16

B.2 Proof of Theorem 1 16

B.3 Proof of Theorem 2 17

C Online Influence Proxy and Staleness Controller **17**

C.1 Online Proxy of A_{ij} 17

C.2 Warm-up calibration of the influence graph 19

C.3 Full sampler with warm-up calibration and integrated controller 19

C.4 Subsampled Newton Method 19

D Additional Experimental Details **21**

D.1 Unconditional Text Generation 21

D.2 Downstream Benchmarks on LLaDA 21

E Additional Experimental Results **24**

E.1 Unconditional Text Generation 24

E.1.1 Entropy-weighted versus uniform decoding 24

E.1.2 Wall-clock cost and controller overhead 24
E.1.3 Effect of controller strength 24
E.1.4 Controller on the capped ReMDM-loop baseline 25
E.2 Downstream Benchmark on LLaDA 25
E.2.1 Reported results for decode-only samplers 25
E.2.2 MATH 26
E.2.3 Hyperparameter ablations 26

F Generated Samples **29**

A Related Work

Discrete and masked diffusion language models. Our work builds on discrete diffusion models (Austin et al., 2021; Campbell et al., 2022; Lou et al., 2023), with a focus on masked diffusion language models. MaskGIT (Chang et al., 2022) introduced iterative parallel decoding from masked states, while MDLM (Sahoo et al., 2024) and recent large-scale models such as LLaDA and Dream (Nie et al., 2025; Ye et al., 2025) show that masked diffusion can scale to language modeling. Recent work further improves decode-only inference through sequence-level energy verification (Xu et al., 2024), entropy-based unmasking (Ben-Hamu et al., 2025), and learned diffusion paths (Peng et al., 2026). These methods mainly operate through forward generation, where revealed tokens are typically not revised.

Training-free remasking and test-time refinement. A closely related line of work introduces inference-time remasking or revocation, allowing revealed tokens to be regenerated. ReMDM (Wang et al., 2025), the most relevant baseline to our work, proposes a training-free remasking sampler with heuristic schedules and demonstrates inference-time scaling. Other training-free decoding rules include backtracking-enhanced remasking for code generation (Dong et al., 2025), self-correction for multimodal masked diffusion (Ouyang et al., 2026), revokable wide-in/narrow-out decoding (Hong et al., 2025), and context-sensitive remasking (Zhai et al., 2026). Some training-free refinement methods also use additional guidance or computation, including reward-guided iterative refinement (Lee et al., 2025), Doob h-transform-inspired process reward modeling for token ordering (Bu et al., 2026), and trajectory-aware entropy steering (Saini et al., 2026). Compared with these methods, BReSC is a direct training-free sampler for pretrained MDMs, without additional reward models or task-specific guidance.

Training-based correction and refinement. Another line of work improves refinement by modifying the model or training objective. Informed correctors (Zhao et al., 2024) learn quantities useful for identifying erroneous tokens, while PRISM (Kim et al., 2025) fine-tunes the model with a self-correction loss to distinguish confidence from correctness. Other methods train auxiliary correction, soft-masking, guidance, anchoring, or path-planning mechanisms (Huang et al., 2025; Schiff et al., 2026; Hersche et al., 2025; Meshchaninov et al., 2025; Rout et al., 2025; Peng et al., 2026). In contrast, BReSC leaves the pretrained MDM unchanged and modifies only the inference sampler.

Predictor-corrector and MCMC perspectives. Correction steps are standard in continuous diffusion, where predictor-corrector samplers use MCMC-like moves to improve intermediate samples (Song et al., 2020). For discrete data, related ideas have been developed through continuous-time Markov-chain formulations (Campbell et al., 2022), informed correctors (Zhao et al., 2024), and discrete flow matching (Gat et al., 2024). For autoregressive models, Karan & Du (2026) use MCMC to sample from sharpened sequence distributions and improve reasoning without retraining. Our work brings the MCMC viewpoint to remasking in MDMs through balanced local transitions and staleness-controlled parallel updates.

Parallel and approximate MCMC. Asynchronous Gibbs sampling studies how stale information affects sampling bias and convergence (De Sa et al., 2016; Terenin et al., 2020). Noisy MCMC

analyzes approximate transition kernels relative to an exact reference (Alquier et al., 2016). Our analysis specializes this perspective to cached parallel remasking: it bounds the discrepancy from a fresh sequential sweep and motivates a controller that reduces interactions between simultaneous updates.

B Proofs and Additional Derivations

B.1 Proof of Proposition 1

Proof. If L_i satisfies coordinate-wise detailed balance, then for every full state $y \in \mathcal{X}$,

$$\sum_x \Pi(x) L_i(x, y) = \sum_x \Pi(y) L_i(y, x) = \Pi(y) \sum_x L_i(y, x) = \Pi(y). \tag{17}$$

Thus each L_i is Π-invariant. A composition of Π-invariant kernels is Π-invariant, so $P = L_n \circ \cdots \circ L_1$ preserves Π. If P is also irreducible and aperiodic on the finite state space $\mathcal{X}$, then the standard finite-state Markov-chain convergence theorem applies. Since Π is invariant and has full support on $\mathcal{X}$, it is the unique stationary distribution of P, and $\delta_x P^m$ converges to Π in total variation for every $x \in \mathcal{X}$. □

B.2 Proof of Theorem 1

Proof. We prove a slightly more general balancing rule and then specialize it to the two targets used by BReSC. Fix a coordinate i, a masked state x with $x_i = \texttt{[MASK]}$, and a token $v \in \mathcal{V}$. The one-site kernel L_i only has nontrivial moves between the masked state x and its revealed neighbor $x_i^{\leftarrow v}$; all other detailed-balance equations are either self-transitions or have zero transition probability in both directions. Thus coordinate-wise detailed balance (9) is equivalent to

$$\Pi(x_i^{\leftarrow v})\, r_i(x_i^{\leftarrow v}) = \Pi(x) d_i(x) p_i(v \mid x, \alpha). \tag{18}$$

Dividing both sides by the common normalizer over the fiber $\{x_i^{\leftarrow a} : a \in \mathcal{V} \cup \{\texttt{[MASK]}\}\}$ gives the conditional form

$$\Pi_i(v \mid x_{-i}) r_i(x_i^{\leftarrow v}) = \Pi_i(\texttt{[MASK]} \mid x_{-i}) d_i(x) p_i(v \mid x, \alpha), \tag{19}$$

which is exactly (10). Whenever $p_i(v \mid x, \alpha) > 0$, this can also be written as

$$d_i(x) = \Gamma_i^{\Pi}(x, v) r_i(x_i^{\leftarrow v}), \qquad \Gamma_i^{\Pi}(x, v) := \frac{\Pi_i(v \mid x_{-i})}{\Pi_i(\texttt{[MASK]} \mid x_{-i})} \frac{1}{p_i(v \mid x, \alpha)}. \tag{20}$$

This is the general design constraint: $d_i(x)$ is a single scalar attached to the masked state, so the right-hand side must be independent of the candidate value v.

For the base target, the ideal-denoiser assumption and independent masking imply

$$\Pi^{\text{base}}_{\alpha,i}(\texttt{[MASK]} \mid x_{-i}) = 1 - \alpha, \qquad \Pi^{\text{base}}_{\alpha,i}(v \mid x_{-i}) = \alpha p_i(v \mid x, \alpha). \tag{21}$$

Hence

$$\Gamma_i^{\Pi^{\text{base}}_\alpha}(x, v) = \frac{\alpha}{1 - \alpha} = \kappa_\alpha, \tag{22}$$

which does not depend on v. Substituting the base policy (12) into (19) gives

$$\Pi^{\text{base}}_{\alpha,i}(v \mid x_{-i}) b = \Pi^{\text{base}}_{\alpha,i}(\texttt{[MASK]} \mid x_{-i}) \kappa_\alpha b p_i(v \mid x, \alpha),$$

so the balance condition holds. A sufficient feasibility condition is $0 \le b \le \min\{1, \kappa_\alpha^{-1}\}$.

For the entropy-tilted target, the cache h is fixed during the balancing step. The tilt contributes no coordinate-i factor when $x_i = \texttt{[MASK]}$, and contributes $e^{-\beta h_i}$ when $x_i = v$. Therefore

$$\frac{\Pi^{\text{ent}}_{\alpha,h,i}(v \mid x_{-i})}{\Pi^{\text{ent}}_{\alpha,h,i}(\texttt{[MASK]} \mid x_{-i})} = \kappa_\alpha e^{-\beta h_i} p_i(v \mid x, \alpha), \qquad \Gamma_i^{\Pi^{\text{ent}}_{\alpha,h}}(x, v) = \kappa_\alpha e^{-\beta h_i}. \tag{23}$$

Using the entropy-balanced policy (13),

$$\begin{aligned}
\Pi^{\text{ent}}_{\alpha,h,i}(v \mid x_{-i}) r_i(x_i^{\leftarrow v}) &= \Pi^{\text{ent}}_{\alpha,h,i}(\texttt{[MASK]} \mid x_{-i}) \kappa_\alpha e^{-\beta h_i} p_i(v \mid x, \alpha) b e^{\beta_R h_i} \\
&= \Pi^{\text{ent}}_{\alpha,h,i}(\texttt{[MASK]} \mid x_{-i}) \kappa_\alpha b e^{-\beta_D h_i} p_i(v \mid x, \alpha) \\
&= \Pi^{\text{ent}}_{\alpha,h,i}(\texttt{[MASK]} \mid x_{-i}) d_i(x) p_i(v \mid x, \alpha),
\end{aligned}$$

where the second equality uses $\beta_R + \beta_D = \beta$. Thus (10) holds for the entropy-tilted target. A sufficient feasibility condition is

$$0 \le b \le \min_{1 \le i \le n} \min\{e^{-\beta_R h_i},\, \kappa_\alpha^{-1} e^{\beta_D h_i}\}.$$

If $p_i(v \mid x, \alpha) = 0$, then the base and entropy-tilted targets also assign zero conditional mass to the revealed value v, so the same balance equations hold trivially. □

B.3 Proof of Theorem 2

Proof. Fix the initial snapshot x. For each coordinate i, define the stale one-site kernel

$$\bar{L}_i^x(s, y) := \ell_i(y_i \mid x)\, \delta_{s_{-i}}(y_{-i}),$$

which updates coordinate i using the law evaluated at the fixed snapshot x, while leaving all other coordinates of the current state s unchanged. Then

$$\delta_x P = \delta_x L_1 L_2 \cdots L_n, \qquad \delta_x \widetilde{P} = \delta_x \bar{L}_1^x \bar{L}_2^x \cdots \bar{L}_n^x,$$

where the products follow the execution order in (6).

By a telescoping expansion and contraction of total variation under Markov kernels,

$$\begin{aligned} \mathrm{TV}\left(P(x,\cdot), \widetilde{P}(x,\cdot)\right) &\le \sum_{i=1}^{n} \left\| \delta_x L_1 \cdots L_{i-1} (L_i - \bar{L}_i^x) \right\|_{\mathrm{TV}} \\ &\le \sum_{i=1}^{n} \mathbb{E}\Big[\mathrm{TV}\left(\ell_i(\cdot \mid X^{(i-1)}), \ell_i(\cdot \mid x)\right)\Big], \end{aligned}$$

where $X^{(i-1)}$ is the fresh-sweep state after the first $i-1$ coordinate updates.

For each i, apply the triangle inequality along the fresh trajectory from $X^{(0)} = x$ to $X^{(i-1)}$:

$$\mathrm{TV}\left(\ell_i(\cdot \mid X^{(i-1)}), \ell_i(\cdot \mid x)\right) \le \sum_{j=1}^{i-1} \mathrm{TV}\left(\ell_i(\cdot \mid X^{(j)}), \ell_i(\cdot \mid X^{(j-1)})\right).$$

Conditioned on $X^{(j-1)} = z$, the jth fresh update draws $V_j \sim \ell_j(\cdot \mid z)$ and sets $X^{(j)} = z_j^{\leftarrow V_j}$. Hence

$$\mathbb{E}\Big[\mathrm{TV}\left(\ell_i(\cdot \mid X^{(j)}), \ell_i(\cdot \mid X^{(j-1)})\right) \mid X^{(j-1)} = z \Big] = A_{ij}(z).$$

Combining the last two displays gives

$$\mathrm{TV}\left(\widetilde{P}(x,\cdot), P(x,\cdot)\right) \le \sum_{1 \le j < i \le n} \mathbb{E}\Big[A_{ij}\big(X^{(j-1)}\big) \Big],$$

which proves the claim. □

C Online Influence Proxy and Staleness Controller

The controller needs a pairwise estimate of how much one token update changes another token's update law. We first derive this estimate from independent cached sweeps, then describe its warm-up calibration and use in the sampler. Throughout this appendix, $A_{ij}(x)$ denotes the exact influence, $A_{ij}^{(1)}(x)$ its first-order approximation, and $\widehat{A}_{t,ij}^{(1)}$ the running estimate. The controller uses W_t, a symmetric, nonnegative normalization of this estimate.

C.1 Online Proxy of A_{ij}

Section 3.3.1 uses a running graph $\widehat{A}_t^{(1)}$ rather than the exact influence matrix $A(x_t)$. We derive an estimator for this graph and identify the higher-order interactions omitted by the approximation.

The exact influence $A_{ij}(x)$ in (14) is not practical to evaluate directly during inference, since it requires counterfactual local-law evaluations after possible updates at coordinate j. Instead, we record which coordinates change during a cached sweep and how much each local update law changes. A Fourier expansion of the resulting function on binary change patterns separates pairwise effects from higher-order interactions (O'Donnell, 2014, Chapter 1).

Fix a state $x \in \mathcal{X}$ and draw a cached proposal $Y \sim \widetilde{P}(x,\cdot)$. For each coordinate m, define

$$\chi_m := \mathbf{1}\{Y_m \neq x_m\}, \qquad \sigma_m(x) := 1 - \ell_m(x_m \mid x), \qquad \Delta_i(x, Y) := \mathrm{TV}\left(\ell_i(\cdot \mid x), \ell_i(\cdot \mid Y)\right).$$

Under the cached product sweep, the change indicators χ_m are independent Bernoulli variables with means $\sigma_m(x)$. For a change pattern c, let

$$\phi_i^x(c) := \mathbb{E}\big[\Delta_i(x, Y) \mid \chi = c, x\big], \qquad c \in \{0,1\}^n.$$

Here $\phi_i^x(c)$ is the expected change in the local law at i, conditional on the change pattern c. We derive the estimator for coordinates with $0 < \sigma_m(x) < 1$; deterministic coordinates are fixed and omitted. Since ϕ_i^x is a function on a finite Boolean cube, it admits the orthogonal expansion under the product Bernoulli measure,

$$\phi_i^x(c) = \sum_{S \subseteq \{1,\dots,n\}} \beta_{i,S}(x) \prod_{m \in S} (c_m - \sigma_m(x)). \tag{24}$$

For each subset S, write the rescaled coefficient as

$$\mathcal{A}_{i;S}(x) := \beta_{i,S}(x) \prod_{m \in S} \sigma_m(x).$$

Theorem 3 (Staleness decomposition and one-sweep estimator)**.** *Fix $i \neq j$ and assume $0 < \sigma_j(x) < 1$. The expected influence in* (14) *decomposes as*

$$A_{ij}(x) = \sum_{r=1}^{n} A_{ij}^{(r)}(x), \qquad A_{ij}^{(r)}(x) := (-1)^{r-1} \sum_{\substack{S \subseteq \{1,\dots,n\}: \\ j \in S,\ |S|=r}} \mathcal{A}_{i;S}(x). \tag{25}$$

In particular, the first-order graph is

$$A_{ij}^{(1)}(x) = \mathcal{A}_{i;\{j\}}(x).$$

Moreover, the one-sweep score

$$Z_{ij}^{(1)}(x, Y) := \Delta_i(x, Y) \frac{\chi_j - \sigma_j(x)}{1 - \sigma_j(x)} \tag{26}$$

satisfies

$$\mathbb{E}\big[Z_{ij}^{(1)}(x, Y) \mid x\big] = A_{ij}^{(1)}(x).$$

Proof. Let $e_j \in \{0,1\}^n$ denote the pattern in which only coordinate j changes. Under the cached product law, conditioning on $\chi = e_j$ makes the new value at j distributed according to the move-only law induced by $\ell_j(\cdot \mid x)$, while all other coordinates remain fixed. Since $\phi_i^x(0) = 0$, the definition in (14) gives

$$A_{ij}(x) = \sigma_j(x)\big(\phi_i^x(e_j) - \phi_i^x(0)\big).$$

Substituting (24), all terms with $j \notin S$ cancel in the difference. For any surviving set S with $j \in S$, the contribution is

$$\sigma_j(x)\, \beta_{i,S}(x) \prod_{m \in S \setminus \{j\}} (-\sigma_m(x)) = (-1)^{|S|-1} \mathcal{A}_{i;S}(x).$$

Grouping terms by $|S| = r$ proves (25).

It remains to show unbiasedness of (26). Taking expectations and using (24),

$$\mathbb{E}\left[\Delta_i(x, Y) \frac{\chi_j - \sigma_j}{1 - \sigma_j} \mid x\right] = \sum_S \beta_{i,S}(x) \mathbb{E}\left[\prod_{m \in S} (\chi_m - \sigma_m) \frac{\chi_j - \sigma_j}{1 - \sigma_j}\right],$$

where we suppress the argument x in σ_m. Independence of the Bernoulli indicators implies that every term vanishes except $S = \{j\}$. For $S = \{j\}$,

$$\mathbb{E}\left[\frac{(\chi_j - \sigma_j)^2}{1 - \sigma_j}\right] = \frac{\sigma_j(1 - \sigma_j)}{1 - \sigma_j} = \sigma_j.$$

Thus the expectation equals

$$\beta_{i,\{j\}}(x)\sigma_j(x) = \mathcal{A}_{i;\{j\}}(x) = A_{ij}^{(1)}(x).$$

□

The estimator in (26) gives a stochastic approximation to the pairwise component of the influence graph under the product cached proposal assumed in the theorem. The higher-order terms in (25) represent multi-coordinate interactions. The controller retains this pairwise graph to keep estimation and optimization tractable during inference.

C.2 WARM-UP CALIBRATION OF THE INFLUENCE GRAPH

The unbiasedness statement in Theorem 3 relies on the proposal being an independent cached product sweep. We therefore separate graph measurement from controlled sampling. Let

$$\mathcal{I}_{\text{loop}} := \{t : \tau_t \in (t_{\text{off}}, t_{\text{on}}]\}$$

be the remasking-loop indices, and let the first

$$K_{\text{warm}} := \lceil \rho_{\text{warm}} |\mathcal{I}_{\text{loop}}| \rceil$$

loop steps be the warm-up steps. During these warm-up steps the controller is deliberately disabled. Given x_{t-1}, we form the balanced one-site laws $\{\ell_{t,i}\}_{i=1}^n$ and sample

$$Y_t \sim \widetilde{P}_t(x_{t-1}, \cdot) := \bigotimes_{i=1}^n \ell_{t,i}(\cdot \mid x_{t-1}), \qquad x_t = Y_t.$$

Since Y_t is sampled from the product sweep, the indicators

$$\chi_{t,i} = \mathbf{1}\{Y_{t,i} \neq x_{t-1,i}\}$$

are conditionally independent Bernoulli variables with means

$$\sigma_{t,i} = 1 - \ell_{t,i}(x_{t-1,i} \mid x_{t-1}).$$

Hence the score in (26) is conditionally unbiased for $A^{(1)}_{ij}(x_{t-1})$ on the warm-up data stream.

On warm-up data, we update the running influence graph by

$$\widehat{A}^{(1)}_{t+1} = (1 - \eta_t)\widehat{A}^{(1)}_t + \eta_t Z^{(1)}_t. \tag{27}$$

The term $\Delta_i(x_{t-1}, Y_t)$ also requires the local law at the proposed state Y_t. The next denoiser evaluation can provide this law, allowing the graph update to use a one-step delay without extra counterfactual model queries.

After warm-up, the controller uses the calibrated graph to sample a dependent change pattern $C \sim \mu_\theta$. These correlated proposals no longer satisfy the independence assumption of Theorem 3, so its unbiasedness guarantee applies only to the warm-up samples. If the implementation continues to update $\widehat{A}^{(1)}$ after control is activated, those post-activation updates are a smoothed tracking heuristic rather than an unbiased estimator under Theorem 3.

C.3 FULL SAMPLER WITH WARM-UP CALIBRATION AND INTEGRATED CONTROLLER

Algorithm 3 is the expanded appendix version of Algorithm 1. It first uses cached product sweeps to estimate the influence graph, then uses the controller to sample dependent update patterns.

C.4 SUBSAMPLED NEWTON METHOD

This subsection gives the implementation details for the solver used in line 3 of Algorithm 2. The key point is that the controller changes only the joint distribution of the binary gate C; once $C_i = 1$, the token value is still sampled from the move-only one-site law q_i in Algorithm 2. Approximate gate calibration can change the realized one-site transition marginals, even though the conditional token law q_i is unchanged.

Proxy-cost guarantee. Fix W_t, σ_t, and $\lambda > 0$, and let $\mu_t^\star$ be an exact minimizer of (16). Since the independent baseline ν_t satisfies the marginal constraints and $D_{\text{KL}}(\nu_t \| \nu_t) = 0$, optimality gives

$$\mathbb{E}_{C \sim \mu_t^\star}[s_t(C)] + \lambda^{-1} D_{\text{KL}}(\mu_t^\star \| \nu_t) \leq \mathbb{E}_{C \sim \nu_t}[s_t(C)]\,.$$

Thus exact optimization does not increase the modeled co-update cost relative to independent sampling, while preserving the prescribed change probabilities. This is a guarantee for the exact proxy optimization, not a guarantee of reduced total-variation discrepancy for the controlled kernel. The finite-sweep implementation approximates this solution; its marginal-calibration diagnostic is described below.

Construction of the controller graph. The controller uses the running first-order graph $\widehat{A}^{(1)}_t$ from Appendix C.1. We first symmetrize and keep only positive estimated interference,

$$B_{t,ij} := \frac{1}{2}\big([\widehat{A}^{(1)}_{t,ij}]_+ + [\widehat{A}^{(1)}_{t,ji}]_+\big), \qquad B_{t,ii} := 0,$$

Algorithm 3 Full BReSC sampler with warm-up calibration and integrated controller

Require: pre-trained denoiser p, length n, timesteps T, time grid $\{\tau_t = (T-t)/T\}$, loop interval $(t_{\text{off}}, t_{\text{on}}]$, controller flag ctrl, warm-up fraction $\rho_{\text{warm}} \in [0,1]$, controller parameters (λ, η_t).

Initialize $x_0 = (\texttt{[MASK]}, \ldots, \texttt{[MASK]})$, entropy cache h, influence estimate $\widehat{A}_0^{(1)} = 0$, dual variable $\theta_0 = 0$, and loop counter $k_{\text{loop}} = 0$.
Set $\mathcal{I}_{\text{loop}} \leftarrow \{t : \tau_t \in (t_{\text{off}}, t_{\text{on}}]\}$ and $K_{\text{warm}} \leftarrow \lceil \rho_{\text{warm}} |\mathcal{I}_{\text{loop}}| \rceil$.
for $t = 1, \ldots, T$ **do**
  Run the denoiser at x_{t-1}; refresh h_i for currently masked sites and hold it fixed during this loop step.
  if $\tau_t \notin (t_{\text{off}}, t_{\text{on}}]$ **then**
    Sample x_t from the standard MDM laws.
  else
    $k_{\text{loop}} \leftarrow k_{\text{loop}} + 1$ and $\mathsf{warm}_t \leftarrow (k_{\text{loop}} \le K_{\text{warm}})$.
    Set the entropy-balanced policy by (13), and form $\ell_{t,i}(\cdot) := \ell_i(\cdot \mid x_{t-1})$ using (4).
    Set $\sigma_{t,i} \leftarrow 1 - \ell_{t,i}(x_{t-1,i} \mid x_{t-1})$ for $i = 1, \ldots, n$.
    if $\neg$ctrl or warm_t **then**
      Sample an independent cached proposal $Y_{t,i} \sim \ell_{t,i}(\cdot)$ for all i, and set $x_t \leftarrow Y_t$.
      if ctrl and warm_t **then**
        Record $\chi_{t,i} = \mathbf{1}\{Y_{t,i} \neq x_{t-1,i}\}$ and $\sigma_{t,i}$; update $\widehat{A}^{(1)}$ with the warm-up score $Z_t^{(1)}$ in (26) and (27).
    else
      Symmetrize and normalize the calibrated graph $\widehat{A}^{(1)}$ to obtain W_t.
      For each $\sigma_{t,i} > 0$, define $q_{t,i}(a) := \ell_{t,i}(a)\mathbf{1}\{a \neq x_{t-1,i}\}/\sigma_{t,i}$; coordinates with $\sigma_{t,i} = 0$ are fixed.
      Using warm-started θ, approximately solve the dual of (16) and sample $C \sim \mu_\theta$ from the binary Gibbs law.

$$\mu_\theta(c) \propto \prod_i \sigma_{t,i}^{c_i}(1-\sigma_{t,i})^{1-c_i} \exp\left\{\theta^\top c - \lambda \sum_{i<j} W_{t,ij} c_i c_j\right\}.$$

      Set $x_{t,i} \leftarrow x_{t-1,i}$ if $C_i = 0$; otherwise sample $Z_{t,i} \sim q_{t,i}(\cdot)$ and set $x_{t,i} \leftarrow Z_{t,i}$.
      Store the updated θ; any graph refresh after this point is a smoothed tracking heuristic, not an unbiased warm-up estimate.
return x_T.

where $[u]_+ = \max\{u, 0\}$. With max normalization, if $M_t := \max_{k<\ell} B_{t,k\ell} > 0$, we set

$$W_{t,ij} = B_{t,ij}/M_t,$$

and otherwise set $W_t = 0$. This gives a symmetric nonnegative matrix with zero diagonal and maximum off-diagonal entry at most one.

Binary Gibbs law. For fixed W_t, σ_t, λ, and dual vector θ, the gate distribution used by the controller is

$$\mu_\theta(c) \propto \prod_{i=1}^n \sigma_{t,i}^{c_i}(1-\sigma_{t,i})^{1-c_i} \exp\left\{\theta^\top c - \lambda \sum_{i<j} W_{t,ij} c_i c_j\right\}, \qquad c \in \{0,1\}^n.$$

Equivalently, for a nondegenerate coordinate $0 < \sigma_{t,i} < 1$, the one-coordinate conditional is

$$\Pr_{\mu_\theta}(C_i = 1 \mid C_{-i}) = \left[1 + \exp\left\{-\left(\log \frac{\sigma_{t,i}}{1-\sigma_{t,i}} + \theta_i - \lambda \sum_{j \neq i} W_{t,ij} C_j\right)\right\}\right]^{-1}.$$

Coordinates with $\sigma_{t,i} = 0$ or $\sigma_{t,i} = 1$ are fixed deterministically and are omitted from the Gibbs and Newton systems.

Blocked Gibbs sampler. We use an 8-block Gibbs implementation. The blocks are deterministic interleaved token classes

$$\mathcal{B}_r = \{i : i \equiv r \pmod 8\}, \qquad r = 0, \ldots, 7.$$

A full Gibbs sweep visits the eight blocks in order. Within a block, all active coordinates are resampled in parallel using the most recent values of coordinates outside the block. If the graph used by the sampler is sparsified and these blocks form a proper coloring, this is exact chromatic Gibbs for that graph. With the dense normalized matrix W_t used in our experiments, it should be viewed as an approximate blocked Gibbs sampler; the correction passes below calibrate the one-site marginals.

Sub-sampled Newton steps. The dual objective $F_t(\theta) = \log Z_t(\theta) - \theta^\top \sigma_t$ from Section 3.3.2 satisfies

$$\nabla F_t(\theta) = \mathbb{E}_{\mu_\theta}[C] - \sigma_t, \qquad \nabla^2 F_t(\theta) = \mathrm{Cov}_{\mu_\theta}(C).$$

A measurement pass runs the 8-block Gibbs sampler and estimates the mean $\widehat{m}$ and covariance $\widehat{H}$ of C from the collected gate samples. A correction pass then applies the sub-sampled Newton-style update

$$\delta = (\widehat{H} + \gamma I)^{-1}(\sigma_t - \widehat{m}), \qquad \theta \leftarrow \theta + 0.5\,\mathrm{clip}(\delta, -5, 5),$$

where clipping is coordinate-wise and $\gamma > 0$ is a small ridge for numerical stability. This is a sub-sampled Newton method: the mean and covariance are estimated from Gibbs samples rather than by enumerating the 2^n gate patterns. In practice, we apply a diagonal approximation to the covariance in this update, using only the estimated marginal variances for computational efficiency.

Initialization, warm starts, and budgets. The first controlled diffusion step uses the mean-field initialization

$$\theta^{\mathrm{mf}}_{t,i} = \lambda \sum_{j \neq i} W_{t,ij}\sigma_{t,j},$$

which cancels the average repulsive field in the conditional logit. Later controlled steps warm-start θ from the previous diffusion step. For OpenWebText, the controller budget is $T_{\mathrm{burn}} = 20$, $T_{\mathrm{measure}} = 200$, $T_{\mathrm{final}} = 20$, and `newton_passes`=2. For each Newton-style pass, we run T_{burn} burn-in Gibbs sweeps, collect T_{measure} measurement sweeps to estimate $\widehat{m}$ and $\widehat{H}$, and apply one correction. After the final correction, we run T_{final} additional Gibbs sweeps and use the last sample as the binary change pattern in Algorithm 2.

Marginal-calibration diagnostic. The exact optimizer of (16) preserves the gate marginals by construction. The practical implementation is approximate because it uses finite Gibbs sampling, blocked updates, and finitely many correction passes. We monitor this approximation by

$$\epsilon_{\mathrm{marg},t} := \max_i \left| \widehat{\mathbb{E}}_{\mu_\theta}[C_i] - \sigma_{t,i} \right|,$$

where $\widehat{\mathbb{E}}_{\mu_\theta}$ is estimated from the measurement samples. We do not make a numerical marginal-error claim unless this diagnostic is explicitly reported.

D Additional Experimental Details

D.1 Unconditional Text Generation

We summarize the OpenWebText unconditional generation setup in Tables 3–5. All methods use the same pretrained checkpoint, tokenizer, sequence length, reference set, and outer sampling schedule. BReSC (no SC) uses the entropy-weighted policy, while BReSC enables the full warm-up/controller schedule in Algorithm 3.

Following the public ReMDM implementation (Wang et al., 2025), we use the outer time grid $t_0, \ldots, t_T = \mathrm{linspace}(1, 10^{-5}, T+1)$ with $\Delta t = (1 - 10^{-5})/T$. The remasking loop is active for $t_{\mathrm{off}} < t \leq t_{\mathrm{on}}$, with $t_{\mathrm{on}} = 0.55$, $t_{\mathrm{off}} = 0.05$, and $\alpha(t_{\mathrm{on}}) = 0.9$. Generation starts from an all-mask sequence of length 1024, does not stop early at EOS, and ends with argmax denoising at the final small timestep. Decoded tokens use nucleus sampling with $p = 0.9$, no top-k truncation, and fp64 Gumbel sampling.

Evaluation protocol. For MAUVE, PPL, and entropy, we use the same OpenWebText evaluation pipeline as the public ReMDM codebase (Wang et al., 2025)[1]. In particular, the reference-set construction, GPT-2 tokenizer decoding, MAUVE feature model, PPL oracle, and empirical token-entropy computation are kept unchanged; our experiments only replace the remasking sampler used to generate the samples. We report the published ReMDM results from Wang et al. (2025) and the published PRISM results from Kim et al. (2025).

D.2 Downstream Benchmarks on LLaDA

We reuse the open-source LLaDA-8B-Instruct model (Nie et al., 2025) for all downstream experiments. Following Wang et al. (2025) and the instructions of their codebase[2], we evaluate LLaDA with

[1] `https://github.com/kuleshov-group/remdm`

[2] `https://github.com/guanghanwang/ReMDM-LLaDA`

Table 3: Main reproducibility settings for OpenWebText unconditional generation.

Item	Setting
Checkpoint	Public MDLM/ReMDM checkpoint `mdlm.ckpt` from the upstream release of Sahoo et al. (2024); no additional training is used.
Model architecture	Small DDiT; sequence length 1024, hidden size 768, 12 transformer blocks, 12 attention heads, conditioning dimension 128, dropout 0.1, untied embeddings.
Tokenizer	HuggingFace GPT-2 tokenizer. GPT-2 vocabulary size is 50257; the diffusion mask token uses `mask_index`$=50257$.
Generation length	Fixed length 1024. Generation starts from the all-mask sequence and does not stop early at EOS. The final residual noise is removed by argmax denoising at the final small timestep.
Sampling budgets	$T \in \{128, 256, 512, 1024\}$. For each method and each T, we generate 5000 samples.
Batch size	Default generation batch size is 20; smaller batch sizes are used when needed to fit memory.
Reference corpus	HuggingFace `openwebtext`, split `train[-100000:]`. We shuffle the validation loader with `valid_seed=42` and use the first 5000 reference blocks.
Reference preprocessing	Documents are tokenized and packed into 1024-token blocks, then decoded back to strings using `tokenizer.batch_decode`. We do not apply additional empty-document filtering, duplicate removal, language filtering, or special-token postprocessing.
Time grid	$t_0, \ldots, t_T = \text{linspace}(1, 10^{-5}, T+1)$ with $\Delta t = (1 - 10^{-5})/T$.
Remasking interval	The remasking loop is active for $t_{\text{off}} < t \leq t_{\text{on}}$ with $t_{\text{on}} = 0.55$, $t_{\text{off}} = 0.05$, and $\alpha(t_{\text{on}}) = 0.9$, following the ReMDM protocol (Wang et al., 2025).
Decoding randomness	Decoded tokens are sampled with nucleus sampling using $p = 0.9$ and no top-k truncation. Categorical samples are drawn using fp64 Gumbel sampling.
Baselines	MDM and ReMDM use the same checkpoint, tokenizer, sequence length, and sampling budgets. PRISM numbers are copied from Kim et al. (2025)

remasking samplers through the lm-evaluation-harness interface. All methods use the same pretrained model and differ only in the inference-time sampler.

For Countdown and TruthfulQA, we follow the short-answer setup of Wang et al. (2025): the maximum generation length is 32, the number of sampling steps is 32, and the block size is 32. For Countdown, we use the 4-shot `countdown3` task and evaluate on 256 synthetic three-number Countdown questions. We generate 10 answers per question and report pass@1 and pass@10 using the task checker. For TruthfulQA, we use the generation benchmark with the fixed six-demonstration prompt (therefore 6-shot) from the ReMDM-LLaDA codebase and evaluate on 817 validation questions. We report ΔROUGE-1, ΔROUGE-2, and ΔROUGE-L (Lin, 2004), computed as the difference between ROUGE against correct answers and ROUGE against incorrect answers. For both Countdown and TruthfulQA, we repeat the experiments with 20 random seeds and report the mean; error bars denote the standard deviation across seeds.

For HumanEval and GSM8K, which require longer generations, we mainly follow the codebase setup of ProSeCo[3] (Schiff et al., 2026). We set up the vanilla LLaDA experiments without additional samplers (ReMDM or BReSC) using the open-source codebase provided by Nie et al. (2025)[4]. The vanilla LLaDA configuration uses temperature-zero token selection. We report point estimates for HumanEval and GSM8K; no statistical significance tests are reported. We use generation length 256, 256 sampling steps, and block size 32, following the block diffusion set-up (Arriola et al., 2025) with the same configuration as in Schiff et al. (2026). HumanEval is evaluated zero-shot in the standard completion format; we use the raw HumanEval prompt without the chat template and truncate completions with the task stop strings before execution. We report pass@1. GSM8K is evaluated with 5 in-context examples (5-shot) on all 1319 test questions. We report exact-match pass@1 under both the flexible extraction and restricted extraction filters from the evaluation harness.

[3] `https://github.com/yair-schiff/proseco`

[4] `https://github.com/ML-GSAI/LLaDA`

Table 4: BReSC and staleness-controller settings for OpenWebText generation.

Item	Setting
BReSC (no SC)	Uses the entropy-weighted policy, but samples all coordinates in parallel. The staleness controller is disabled.
BReSC	Uses the same policy as BReSC (no SC), but enables the full warm-up/controller schedule in Algorithm 3 during the remasking loop.
Entropy target parameter	$\beta_D = 1/8 = 0.125$, $\beta_R = 1.0$ and $\beta = 1.125$. The code uses decode temperature $\tau_D = 8$, with $\beta_D = 1/\tau_D$.
Controller strength	$\lambda = 50$.
Warm-up fraction	$\rho_{\text{warm}} = 0.3$.
Remasking budget scale	$\sigma_{\max} = \min\{(1-\alpha)/\alpha,\ n_M/\max(n_U, 1),\ 1\}$, where n_M, n_U count masked and revealed positions. At $\alpha = 0.9$, the nominal ceiling is $1/9 \approx 0.111$.
Controller activation	The controller is active only inside the remasking loop and only after a 30% loop warm-up defined by ρ_{warm}. This corresponds to 19, 38, 77, and 154 warm-up steps for $T = 128, 256, 512, 1024$, respectively.
Weight normalization	Max normalization.
Binary sampler	The gate distribution is sampled with the 8-block Gibbs routine described in Appendix C.4; one full sweep updates all eight interleaved blocks.
Dual calibration	Warm-started sampled Newton calibration with mean-field initialization at the first controlled step; 2 measure/correct passes; coordinate-wise correction clipping to ± 5; damping factor 0.5.
Controller budget	For each measure/correct pass, `T_burn`$= 20$ burn-in sweeps and `T_measure`$= 200$ measurement sweeps; after the final correction, `T_final`$= 20$ sweeps; `newton_passes`=2.

Table 5: Evaluation and implementation details for OpenWebText generation.

Item	Setting
Evaluation protocol	Same OpenWebText metric-computation pipeline as the public ReMDM implementation: reference-set construction, GPT-2 tokenizer decoding, MAUVE computation, PPL computation, and empirical token entropy are kept unchanged.
MAUVE	Computed using `mauve-text` with GPT-2-large features, maximum text length 1024, batch size 8, device id 0, and seed 42. MAUVE compares the 5000 generated strings against the same 5000 OpenWebText reference strings.
PPL	Computed using HuggingFace `gpt2-large`. Generated samples are re-tokenized to maximum length 1024. We compute token-count-weighted cross-entropy over generated samples only and then exponentiate. PPL batch size is 8.
Entropy	Mean per-sequence empirical unigram token entropy, in nats, computed over fixed-length generated token rows.
Hardware	Experiments are conducted on NVIDIA H100 80GB GPUs.
Software	Python 3.9, CUDA toolkit `cuda-nvcc` 12.4.99 with `pytorch-cuda` 12.1, Transformers 4.38.2, Datasets 2.18.0, and `mauve-text` 0.4.0.

To reduce wall-clock time, GSM8K is sharded across multiple RTX A6000 GPUs and merged before scoring.

For ReMDM, we use the LLaDA-adapted ReMDM-loop sampler of Wang et al. (2025) in the open-source codebase[5]. Specifically, for each block of 32 tokens, whenever 28 tokens have been decoded, we will enable the ReMDM loop: for another 32 steps we remask k tokens that had the lowest confidence score and unmask k tokens based on their confidence. For Countdown and TruthfulQA, we follow Wang et al. (2025) and set $k = 1$ and $k = 4$ respectively. For HumanEval and GSM8K, we follow the choice in Schiff et al. (2026) and set $k = 2$ for both tasks.

For our BReSC, we replace the confidence-based remasking rule with our entropy-balanced remasking rule. The main hyperparameters of our sampler are the proposal-support threshold q (which truncates

[5] `https://github.com/guanghanwang/ReMDM-LLaDA`

the tail of p_i when computing the entropy h_i) and the remasking capacity k (the same role as in ReMDM above). We use $q = 0.85, k = 3$ for Countdown, $q = 0.75, k = 6$ for TruthfulQA, $q = 0.70, k = 1$ for HumanEval, and $q = 0.65, k = 3$ for GSM8K.

Hyperparameter selection protocol. The reported q and k values were selected using benchmark-specific evaluations. Tables 12–15 report local sensitivity around the selected configurations, varying one parameter while holding the other fixed. These tables document local robustness rather than held-out hyperparameter selection.

All downstream experiments are run on NVIDIA RTX A6000 GPUs.

E Additional Experimental Results

E.1 Unconditional Text Generation

E.1.1 Entropy-weighted versus uniform decoding

To assess the contribution of the decoding-rate design beyond entropy-based remasking scores, we retain the same entropy-based remasking scores and replace the entropy-weighted decoding rates with a uniform rate across masked positions. The staleness controller is disabled in both arms. We evaluate the two policies with 5000 samples at each budget $T \in \{16, 32, 64, 128\}$. Table 6 reports both arms from this dedicated ablation, separately from the main comparison in Table 1.

At $T = 128$, BReSC (no SC) improves PPL from 29.5 to 27.2 and MAUVE from 0.120 to 0.146 relative to uniform decoding. Both metrics also improve at $T = 64$ and $T = 32$. At $T = 16$, both policies produce poor samples and have the same reported MAUVE, indicating that neither is effective at this very small budget. These results support the contribution of the paired remasking and decoding-rate design beyond the remasking scores alone.

Table 6: Entropy-weighted versus uniform decoding on OpenWebText, with 5000 samples per policy and budget. Both policies use the same entropy-based remasking scores and disable staleness control. The uniform-decode variant assigns the same decoding rate to every masked position.

	PPL ↓		MAUVE ↑	
T	BReSC (no SC)	Uniform decode	BReSC (no SC)	Uniform decode
128	**27.2**	29.5	**0.1460**	0.1200
64	**43.2**	47.4	**0.0245**	0.0209
32	**97.4**	107.1	**0.0063**	0.0057
16	610.5	610.6	0.0044	0.0044

E.1.2 Wall-clock cost and controller overhead

We separately measure generation time per sample on the same NVIDIA A100 GPU at $T = 1024$, averaging over 20 samples for each method. Table 7 reports seconds per sample for ReMDM, BReSC without staleness control, and full BReSC including its controller. Without the controller, BReSC adds approximately 9% to the generation time per sample relative to ReMDM (16.2 versus 14.8 seconds per sample). Full BReSC takes 27.3 seconds per sample, or approximately $1.84\times$ the ReMDM time at the same sampling budget. These measurements quantify per-sample overhead at equal T; the sampling-step reductions in Figure 2 are a separate comparison. Combining the approximately fivefold reduction in steps at matched MAUVE with the measured overhead gives $5/(27.3/14.8) \approx 2.7$ as an estimated wall-clock speedup. This extrapolation assumes that generation time scales linearly with T and that the relative overhead remains comparable at smaller budgets; it is not a direct timing measurement at matched quality.

E.1.3 Effect of controller strength

We first ablate the strength of the repulsive controller by sweeping the regularization parameter λ at $T = 128$. As shown in Table 8, adding the controller improves PPL slightly across all tested values of λ, while MAUVE is more sensitive to the amount of repulsion. Moderate repulsion gives

Table 7: Generation time per sample on an NVIDIA A100 at $T = 1024$, averaged over 20 samples per method. Full BReSC includes the controller cost. Relative time is normalized to ReMDM.

Method	Seconds/sample	Relative time
ReMDM	14.8	1.00×
BReSC (no SC)	16.2	1.09×
BReSC	27.3	1.84×

the best generation quality: $\lambda = 50$ achieves the highest MAUVE, improving from 0.1264 without control to 0.1519. Increasing λ to 100 further lowers PPL but reduces MAUVE, suggesting that overly aggressive suppression of co-updates can remove useful parallel refinement. This supports our view of λ as a trust-region knob: it controls how much joint-dependence budget is spent to reduce stale-context interference.

Table 8: Ablation of the controller strength λ at $T = 128$. PPL is lower-is-better and MAUVE is higher-is-better. Best results are in **bold**; second best are underlined.

Setting	PPL ↓	MAUVE ↑
No control	29.06	0.1264
Control ($\lambda = 20$)	28.49	0.1265
Control ($\lambda = 30$)	28.49	<u>0.1363</u>
Control ($\lambda = 50$)	<u>28.48</u>	**0.1519**
Control ($\lambda = 100$)	**28.41**	0.1335

E.1.4 Controller on the capped ReMDM-loop baseline

We next test whether the same controller improves a capped ReMDM-loop baseline. Here we use $\eta = 0.04$ and compare the uncontrolled loop against the controlled loop with $\lambda = 100$. Table 9 shows that the controller improves both PPL and MAUVE at short and medium horizons ($T = 128, 256, 512$). The improvement is especially pronounced at $T = 128$, where MAUVE increases from 0.018 to 0.200, indicating that repulsive control is most useful when the sampler has a limited refinement budget and stale co-updates are more damaging. At $T = 1024$, the uncontrolled baseline slightly outperforms the controlled version, suggesting that very long refinement horizons can partially correct stale updates on their own, while a strong controller may become unnecessarily conservative. The entropy values remain nearly unchanged, which is consistent with the design of the gate: it reshapes the joint coupling of updates rather than changing the one-site entropy policy.

Table 9: Ablation of the repulsive controller on the capped ReMDM-loop baseline (Algorithm 3 in Wang et al. (2025)) with $\eta = 0.04$. PPL is lower-is-better and MAUVE is higher-is-better; entropy is a diversity diagnostic. The best PPL and MAUVE within each T are in **bold**.

	PPL ↓				MAUVE ↑				Entropy			
Method / $T =$	128	256	512	1024	128	256	512	1024	128	256	512	1024
No control	67.160	49.210	37.910	**27.700**	0.018	0.059	0.214	**0.355**	5.57	5.52	5.45	5.32
Control ($\lambda = 100$)	**66.280**	**49.080**	**37.800**	28.600	**0.200**	**0.075**	**0.224**	0.351	5.57	5.51	5.44	5.35

E.2 Downstream Benchmark on LLaDA

E.2.1 Reported results for decode-only samplers

Training-free methods also accelerate generation without remasking. Fast-dLLM combines approximate KV caching with confidence-aware parallel unmasking (Wu et al., 2026), while KLASS uses confidence and temporal prediction stability to select parallel updates (Kim et al., 2026). These methods focus on efficient decoding; our study focuses on revising previously generated tokens within a remasking correction loop and coordinating updates within that loop.

Table 10 collects reported results using LLaDA-8B-Instruct and a generation length of 256 tokens. The external results are taken from the original papers, without rerunning their methods. Block sizes, inference costs, evaluation details, and hyperparameter-selection procedures differ, so these scores provide context but do not establish a ranking under matched conditions.

Table 10: Reported downstream scores (%) for LLaDA-8B-Instruct at generation length 256. HumanEval reports pass@1. Our GSM8K score uses flexible matching; external GSM8K scores follow their source evaluations. Inference costs and evaluation protocols are not matched.

Method	Block size	HumanEval	GSM8K
Fast-dLLM (cache + parallel) (Wu et al., 2026)	32	43.3	78.5
KLASS (Kim et al., 2026)	64	40.85	76.50
LLaDA+BReSC (Table 2)	32	46.3	80.4

The corresponding vanilla LLaDA HumanEval/GSM8K scores are 39.6/77.3 in our evaluation, 41.5/79.3 in Fast-dLLM, and 39.63/75.13 in KLASS, illustrating the variation between evaluation setups.

E.2.2 MATH

We additionally evaluate the three LLaDA-8B-Instruct samplers on MATH (Hendrycks et al., 2021). Table 11 reports accuracy: LLaDA+BReSC reaches 34.08%, compared with 32.92% for LLaDA and 33.18% for LLaDA+ReMDM. This corresponds to gains of 1.16 and 0.90 percentage points over LLaDA and LLaDA+ReMDM, respectively.

Table 11: Additional MATH evaluation with LLaDA-8B-Instruct. We report accuracy in percentages.

Method	Accuracy (%) ↑
LLaDA	32.92
LLaDA+ReMDM	33.18
LLaDA+BReSC	**34.08**

E.2.3 Hyperparameter ablations

Tables 12–15 assess local sensitivity around the selected hyperparameters; Appendix D.2 describes the selection protocol. For Countdown, varying $k = 1, \ldots, 5$ at $q = 0.85$ gives pass@1 scores of 50.55–51.09%, consistently above both baselines in Table 2. Increasing q improves pass@10 but can reduce pass@1. For HumanEval, all nine displayed configurations exceed both reported baselines: pass@1 ranges from 42.07 to 46.34%, versus 39.6% for LLaDA and 39.0% for ReMDM. For GSM8K, all nine displayed configurations exceed ReMDM under restricted extraction (44.96–49.36% versus 40.3%), whereas flexible-extraction gains are less consistent. TruthfulQA is more sensitive, with the best displayed settings near $q = 0.70$–0.75 and $k = 6$. These ablations show that several improvements extend beyond the selected configuration, particularly on HumanEval and Countdown; they establish local robustness rather than held-out hyperparameter selection.

Table 12: Countdown ablation around the best configuration. We report pass@1 and pass@10 in percentages.

q	k	pass@1	pass@10
0.85	1	50.94	64.06
0.85	2	50.94	63.67
0.85	3	**51.09**	64.06
0.85	4	50.55	63.67
0.85	5	50.66	63.28
0.75	3	48.36	51.56
0.80	3	48.20	54.30
0.90	3	47.03	75.00
0.95	3	37.89	**79.30**

Table 13: TruthfulQA single-seed ablation around the best configuration. We report ΔROUGE-1/2/L and their average.

q	k	ΔROUGE-1	ΔROUGE-2	ΔROUGE-L	Avg.
0.75	3	26.25	29.45	26.40	27.36
0.75	4	26.19	29.49	26.31	27.33
0.75	5	27.74	30.93	27.94	28.87
0.75	6	**28.45**	**31.06**	**28.72**	**29.41**
0.75	7	25.54	27.07	25.93	26.18
0.65	6	27.29	29.58	27.37	28.08
0.70	6	**28.45**	30.85	28.42	29.24
0.80	6	25.70	27.94	25.83	26.49
0.85	6	26.32	28.13	26.57	27.01
0.90	6	22.75	24.14	22.95	23.28

Table 14: HumanEval single-seed ablation around the best configuration. We report pass@1 in percentages.

q	k	pass@1
0.70	1	**46.34**
0.70	2	45.12
0.70	3	45.12
0.70	4	45.73
0.70	5	45.73
0.65	1	43.90
0.75	1	42.68
0.80	1	43.29
0.85	1	42.07

Table 15: GSM8K ablation around the best configuration. We report flexible and restricted pass@1 in percentages.

q	k	flexible pass@1	restricted pass@1
0.65	1	78.39	48.90
0.65	2	78.24	47.31
0.65	3	**80.44**	48.60
0.65	4	78.47	**49.36**
0.65	5	78.62	45.34
0.60	3	79.38	49.20
0.70	3	79.53	46.10
0.75	3	78.47	45.34
0.80	3	78.01	44.96

F GENERATED SAMPLES

We show several representative cases where BReSC gives the correct answer while both vanilla LLaDA and ReMDM make the same or similar error.

GSM8K: quantity tracking. **Question.** John buys twice as many red ties as blue ties. The red ties cost 50% more than blue ties. He spent $200 on blue ties that cost $40 each. How much did he spend on ties?

Gold answer. 800.

LLaDA. 1400. The model treats John as buying 10 blue ties, although $200 at $40 each gives only 5 blue ties.

LLaDA+ReMDM. 1400. The same counting error is made before computing the red-tie cost.

LLaDA+BReSC. 800. The model computes 5 blue ties, 10 red ties, $600 spent on red ties, and $800 total.

GSM8K: interpreting round trips. **Question.** Blake and Kelly are having a contest to see who can run the most in 15 minutes. They decide to do it on a football field that is 100 yards long. Blake runs back and forth 15 times. Kelly runs back and forth once, and then runs to the 40-yard line and back 34 times. How much farther does the winner run than the loser?

Gold answer. 80.

LLaDA. 1320. The model counts Blake's "back and forth" distance as 15×100 rather than 15×200.

LLaDA+ReMDM. 1320. The same round-trip error leads to the wrong winner margin.

LLaDA+BReSC. 80. The model correctly computes Blake's distance as 3000 yards and Kelly's as 2920 yards.

HumanEval: base conversion. **Task.** `HumanEval/44`, `change_base(x, base)`. Convert an integer to a string representation in a base below 10.

LLaDA. `result.append(str(remainder)); return ''.join(result)`. The digits are appended in least-significant-first order, so the output is reversed.

LLaDA+ReMDM. `result.append(str(remainder)); return ''.join(result)`. ReMDM makes the same ordering mistake.

LLaDA+BReSC. `result = str(remainder) + result`. The generated solution prepends each new digit and passes the tests.

HumanEval: sorting by a compound key. **Task.** `HumanEval/116`, `sort_array(arr)`. Sort non-negative integers by the number of ones in their binary representation, breaking ties by the integer value.

LLaDA. `return sorted(arr, key=lambda x: (x, bin(x).count('1')))`. The model sorts primarily by the integer value.

LLaDA+ReMDM. `return sorted(arr, key=lambda x: (x, bin(x).count('1')))`. The same incorrect key order is used.

LLaDA+BReSC. `return sorted(arr, key=lambda x: (bin(x).count('1'), x))`. The generated solution uses the specified primary and secondary keys.