A STEP IN THE RIGHT PLACE: UNDERSTANDING AND OPTIMIZING TIMESTEP PLACEMENT IN CONTINUOUS DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous diffusion language models represent an emerging paradigm for language modeling. They generate text in parallel by iteratively refining continuous representations and mapping them back to tokens through a decoder. However, realizing the efficiency advantage of parallel generation requires maintaining generation quality with only a few denoising steps. Even when the number of steps is fixed, their placement along the denoising trajectory remains flexible. In this paper, we demonstrate that changing timestep placement alone can substantially alter the final latent representations and thereby affect the decoded token distributions. Building on these findings, we introduce **StepTune**, an algorithm that searches for timestep placements that improve generation quality. The search is conducted on a small calibration set and guided by a language-aware objective, with the model and sampling budget fixed. Once calibrated, the timestep placement is fixed and reused across inputs. Experiments demonstrate consistent improvements in unconditional generation perplexity and conditional task metrics across all evaluated sampling budgets. In particular, StepTune enables 16-step sampling with a frozen model to outperform its native 64-step baseline on both XSum summarization and WMT14 translation and reduces end-to-end latency by approximately $3.8\times$. These results establish timestep placement as a practical means of improving the quality–efficiency trade-off in continuous diffusion language models.

1 INTRODUCTION

Autoregressive language models generate text one token at a time, conditioned on the preceding sequence (Radford et al., 2019). Diffusion language models offer an alternative by iteratively denoising a sequence, allowing multiple positions to be updated in parallel (Li et al., 2022; Sahoo et al., 2024). Within this paradigm, continuous diffusion language models (continuous DLMs), such as ELF (Hu et al., 2026) and AURORA (Liang et al., 2026), map tokens into continuous representations, perform denoising in this space, and decode the resulting representations back into tokens. However, generating a sequence in parallel still requires sequential rounds of refinement. Realizing the efficiency advantage of parallel generation therefore requires maintaining generation quality with only a few denoising steps, measured by the number of function evaluations (NFE).

At a limited NFE budget, the placement of denoising steps along the denoising trajectory remains flexible. With the model and solver unchanged, different timestep placements allocate computation differently between the initial noise and the final representation. Prior work on image diffusion has demonstrated the benefits of optimizing this allocation (Sabour et al., 2024; Tong et al., 2025). For language generation, we seek to determine whether timestep placement offers similar benefits for token distributions and generation quality. This raises two central questions:

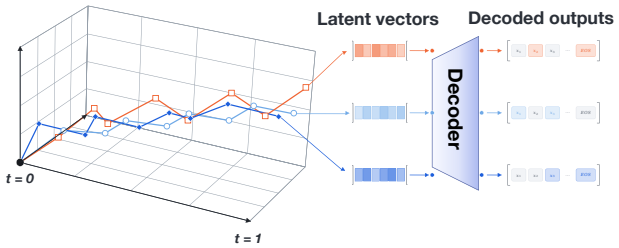


Figure 1: **Timestep placement shapes generation.** Different timestep placements alter the sampling trajectory, eventually leading to different decoded outputs.

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.

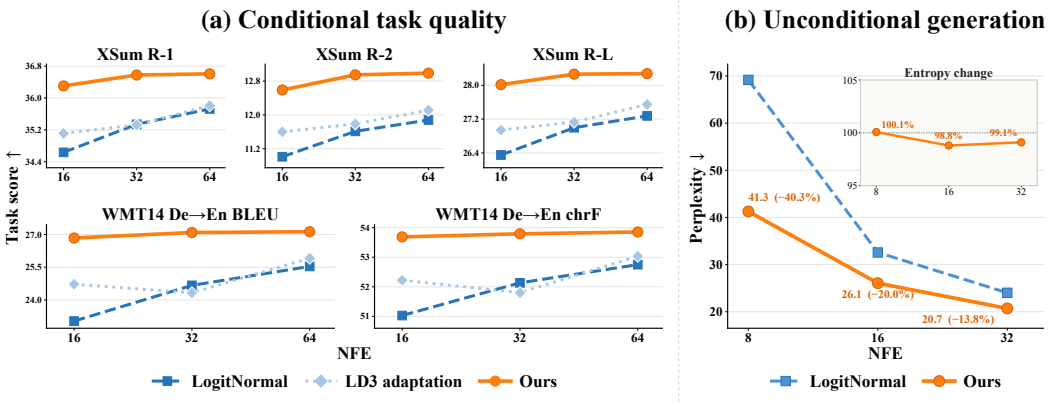


Figure 2: **StepTune improves generation quality with fewer denoising steps.** (a) On XSum summarization and WMT14 De→En translation, 16-step sampling outperforms the native 64-step baseline. (b) Unconditional generation with the official ELF SDE sampler shows consistently lower perplexity at matched NFE. The inset reports unigram entropy relative to LogitNormal (100%).

(i) How much control does timestep placement provide over the final token distributions? (ii) How can we use this control to improve generation at a fixed NFE budget?

To address question (i), we empirically investigate the effects of timestep placement in Section 3. We first demonstrate that changing timestep placement alone can substantially alter the final latent representations, which the decoder subsequently maps to token distributions (Figure 1). To better assess differences in token distributions, we use a high-NFE trajectory from the same model as a numerical reference. It uses the same input and initial noise but takes finer denoising steps to obtain a more accurate numerical solution.

Using this reference, our analysis in Section 3.2 reveals two distinct regimes. When latent deviations from the reference are large, latent error is strongly positively correlated with the KL divergence between decoded token distributions: smaller latent errors tend to correspond to smaller KL. As the latent representations approach the reference, the correlation between latent error and KL drops sharply. Because the decoder's sensitivity to error direction becomes important at small scales, latent distance alone no longer adequately captures deviations in decoded token distributions.

These findings provide a bridge from question (i) to question (ii). Finding improved timestep placements is a search problem with infinitely many possibilities in a continuous space, so an objective is needed to guide the search. Euclidean and perceptual distances to reference outputs have been used as optimization objectives in image diffusion sampling (Shaul et al., 2024; Tong et al., 2025; Frankel et al., 2025). For language generation, however, image-specific perceptual metrics such as LPIPS (Zhang et al., 2018) are not directly applicable. Our analysis also shows that latent distance alone cannot adequately capture changes in token distributions.

We therefore draw on the complementary behavior observed above to construct a joint discrepancy. Intuitively, the latent term encourages movement toward the reference, providing guidance at larger deviations where latent error and KL are strongly correlated. Closer to the reference, the KL term distinguishes changes in token probabilities that latent distance alone cannot capture. Together, these terms favor representations that are close to the reference and yield similar token distributions. For timestep placement θ, let $A(\theta)$ denote latent error and $K(\theta)$ denote decoder KL. With positive weights w_A and w_K, we define

$$F(\theta) = w_A A(\theta) + w_K K(\theta). \tag{1}$$

Building on this objective, we now turn to question (ii) and introduce **StepTune**, an algorithm that searches for improved timestep placements on a small calibration set with the model, solver, and NFE fixed. StepTune proceeds in three stages (Section 4). First, we compress the space of timestep placements through a compact parameterization of step allocation or a finite set of candidate placements to make the search tractable. Second, we use F to guide the search toward regions with the potential to improve generation quality, subject to constraints evaluated on the calibration set. Third, we select the final timestep placement based on calibration results and freeze it for reuse.

By changing only timestep placement, StepTune delivers consistent improvements in generation quality. On ELF (Hu et al., 2026), it improves unconditional generation perplexity and conditional task metrics across all evaluated NFE budgets (Figure 2). In particular, 16-step sampling outperforms the native 64-step baseline on both XSum summarization (Narayan et al., 2018) and WMT14 translation (Bojar et al., 2014), achieving an approximately $3.8\times$ end-to-end speedup with the same frozen model. These gains are not limited to a single continuous DLM architecture. On AURORA (Liang et al., 2026), StepTune also enables 16-step sampling to outperform the native 32-step baseline on XSum summarization. For unconditional generation, it reduces perplexity by 26.9% at the same 16-step budget. Together, these results demonstrate that optimizing timestep placement can improve both generation quality and sampling efficiency across distinct continuous DLM architectures.

Our main contributions are threefold.

1. **Understanding timestep placement.** We show how timestep placement alters terminal representations and why latent distance alone can fail to capture changes in decoded token distributions.
2. **Language-aware calibration.** We introduce StepTune to optimize timestep placement using F under quality constraints and freeze the selected schedule for reuse across inputs.
3. **Improved quality and efficiency.** We demonstrate gains across continuous DLM architectures, achieving better generation quality with fewer denoising steps and no model updates.

2 Related Work

Diffusion language models. Diffusion language models generate text through iterative denoising. Discrete DLMs define corruption and denoising directly in token space (Austin et al., 2021; Sahoo et al., 2024; Nie et al., 2025), whereas continuous DLMs operate on continuous token representations and decode the resulting representations into text (Li et al., 2022; Dieleman et al., 2022; Gulrajani & Hashimoto, 2023; Hu et al., 2026). More closely related work studies training noise schedules and inference timestep placement. LangFlow learns a Gumbel noise schedule motivated by information-uniform denoising (Chen et al., 2026), RePlaid learns its noise schedule by reducing ELBO-estimator variance (Yang et al., 2026), and ELF demonstrates the benefits of logit-normal over uniform inference times (Hu et al., 2026). Whereas these studies learn training schedules or compare predefined inference rules, we study inference timestep placement as the sole variable. We optimize it directly, with all other settings fixed, and our method has no hard dependency on any predefined schedules.

Optimizing timestep placement. Prior work has directly optimized inference timestep placement in visual diffusion. AYS derives its objective from a bound on sampling-distribution divergence (Sabour et al., 2024), while GITS uses dynamic programming to select timesteps based on trajectory structure (Chen et al., 2024). LD3 learns timesteps by matching the outputs of a low-NFE sampler to a high-NFE reference (Tong et al., 2025). S4S-Alt also uses reference matching but jointly optimizes timesteps and solver coefficients (Frankel et al., 2025). OYS instead uses Bayesian optimization to search directly against image-quality metrics (Zhang et al., 2026). In contrast, we study timestep placement in continuous DLMs. We investigate the language-specific relationship between changes in terminal representations and the resulting changes in decoded token distributions. Building on this analysis, we develop a joint objective and a calibration algorithm to improve language generation. We provide an extended discussion of additional related work in Appendix D.

3 Understanding Timestep Placement

In this section, we seek to understand the role of timestep placement in continuous DLM generation. We empirically examine how it changes terminal latent representations and how these changes propagate through the decoder to affect token distributions. We further test whether the effects of a calibrated timestep placement persist across held-out inputs.

Preliminaries. Continuous DLMs use an encoder to represent text in a continuous latent space and a decoder to map latent representations back to token probabilities (Hu et al., 2026; Liang et al., 2026). Let t denote progress along the sampling process during inference. Sampling starts from Gaussian noise at $t = 0$ and progressively transforms it into a latent representation at $t = 1$ through

finite denoising steps. At step k, let z_k and z_{k+1} denote the latent states before and after an update from time t_k to t_{k+1}. Let v_k denote the velocity computed from the model's prediction at the current state and time. A typical Euler update takes the form

$$z_{k+1} = z_k + (t_{k+1} - t_k)v_k. \tag{2}$$

Each Euler update treats the predicted velocity as constant over the time interval. Multiplying this velocity by the interval length approximates the change in the latent state. Sampling thus approximates the continuous denoising process through discrete updates over a finite sequence of time intervals. Timestep placement determines how these intervals are distributed along the sampling process. We denote the terminal latent representation by Z. The decoder converts Z into token probabilities from which the output text is generated. Appendix A provides a theoretical account of the full sampling and decoding process, while the following subsections present empirical and illustrative analyses.

3.1 Timestep Placement Shapes Terminal Representations

Intuitively, different timestep placements can lead to different terminal representations even at the same NFE. But how large can this effect become in a continuous DLM?

We test this effect using ELF-B on 128 held-out XSum inputs at NFE–16. We gradually shift the timesteps from a uniform placement ($\lambda = 0$) to a fixed nonuniform placement ($\lambda = 1$), keeping each input's initial noise unchanged. Let Z_λ denote the resulting terminal representation and Z_U the uniform baseline's terminal representation. We measure their relative terminal distance as $\|Z_\lambda - Z_U\|_2 / \|Z_U\|_2$ over the generated positions.

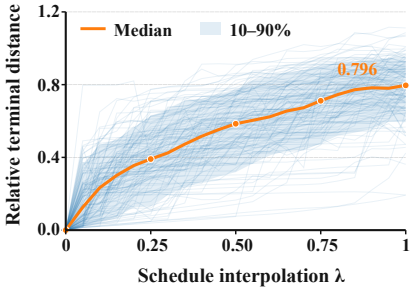


Figure 3: **Timestep placement changes terminal representations.** Relative displacement from uniform sampling.

Figure 3 shows that, at $\lambda = 1$, the median terminal displacement reaches 79.6% of the uniform endpoint's norm. Even at $\lambda = 0.1$, moving the timesteps only one tenth of the way toward the nonuniform placement already produces a median displacement of 23.3%. **Timestep placement alone can substantially change the final latent representations at a fixed NFE budget.**

3.2 From Latent Error to Token Divergence

We next examine how terminal latent shifts affect decoded token distributions. Since finer sampling steps more accurately approximate the continuous generation process, we use a high-NFE run as a reference under the same model, solver, input, and initial noise. Let $Z^\star$ denote the reference endpoint, Z a sampled endpoint, and $q(Z)$ the decoded token probabilities. The latent error is $e = Z - Z^\star$. A matrix W_{dec} describes the decoder's local sensitivity to different error directions, giving the *decoder-aware quadratic approximation*

$$\mathrm{KL}(q(Z^\star) \,\|\, q(Z^\star + e)) \approx \frac{1}{2} e^\top W_{\text{dec}} e. \tag{3}$$

This expression depends on both error magnitude and direction. The derivation is provided in Appendix A.

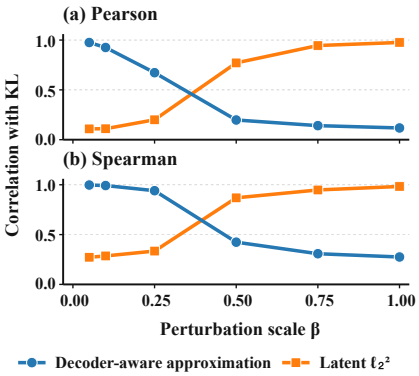


Figure 4: **Correlations with KL across six perturbation scales.**

We test this relationship using 96 terminal-error directions from NFE–16 schedules on XSum, with an NFE–64 reference. Along each direction, we evaluate the decoder at $Z^\star + \beta e$ for six scales from $\beta = 0.05$ to 1. At each scale, we compare how strongly normalized squared latent distance and the quadratic approximation correlate with the actual KL divergence.

Figure 4 reveals two distinct regimes. At the largest scale, latent distance is strongly positively correlated with KL: smaller latent errors tend to accompany smaller deviations in token distributions. Near the reference, however, this relationship weakens sharply. At $\beta = 0.05$ and 0.1, the

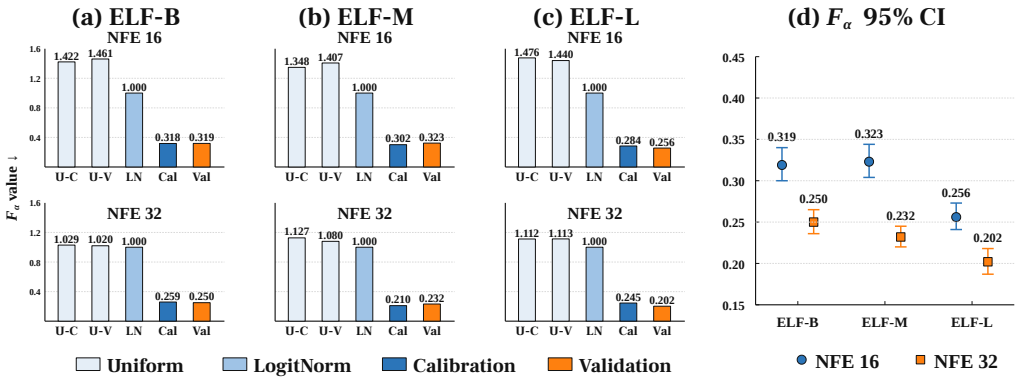


Figure 5: **Joint discrepancy reductions persist beyond calibration.** (a–c) Frozen schedules reduce normalized F_α on calibration and unseen validation trajectories (LogitNormal $= 1$). (d) Validation scores with 95% confidence intervals.

Pearson correlation between latent distance and KL drops to about 0.11, whereas the decoder-aware approximation achieves 0.975 and 0.925. Spearman correlations show the same pattern.

The quadratic approximation helps explain why latent distance alone is insufficient near the reference: latent distance measures error magnitude but ignores direction, whereas the approximation accounts for the decoder's sensitivity to different error directions. Its weaker correlation with KL at larger scales is expected, since it describes local behavior around the reference.

Latent distance is strongly correlated with KL at larger deviations, but near the reference, latent distance alone cannot adequately capture deviations in decoded token distributions.

3.3 A Joint Objective for Timestep Calibration

The preceding analysis examines how timestep placement affects terminal representations and decoded token distributions. Our goal, however, is to find timestep placements that improve generation quality. Finding such placements in a continuous space is a search problem, which our algorithm in Section 4 addresses. To guide this search under a limited calibration budget, we need an objective that provides useful guidance and whose reductions persist beyond the calibration trajectories.

Our primary interest is agreement with the reference token distributions, measured by decoder KL. Our preceding analysis shows that, at larger deviations, smaller latent errors tend to correspond to smaller KL. Near the reference, however, latent distance alone cannot adequately capture differences in token probabilities because of the decoder's sensitivity to error direction.

We therefore formulate a joint objective that accounts for both latent error and decoder KL. Latent error provides a direct signal of endpoint deviation before it is transformed by the decoder, offering auxiliary guidance toward the reference, particularly at larger deviations. Decoder KL captures the actual differences in token probabilities, including those that latent distance alone misses near the reference. Section 5.3 compares this objective with using latent error or decoder KL alone.

For a schedule θ, let $A(\theta)$ and $K(\theta)$ denote the mean relative squared latent error and mean decoder KL, respectively, over the evaluation trajectories. Using positive baseline normalization constants A_{scale} and K_{scale} and a weight $\alpha \in [0, 1]$, we define

$$F_\alpha(\theta) = \alpha \frac{A(\theta)}{A_{\text{scale}}} + (1 - \alpha) \frac{K(\theta)}{K_{\text{scale}}}. \tag{4}$$

We set $\alpha = 1/2$; normalization details are provided in Appendix B.

We test whether calibration with this objective substantially reduces both latent error and, more importantly, decoder KL on unseen trajectories across model scales and NFE budgets. We calibrate one schedule for NFE–16 and another for NFE–32 using 16 trajectories from ELF-M. Both schedules

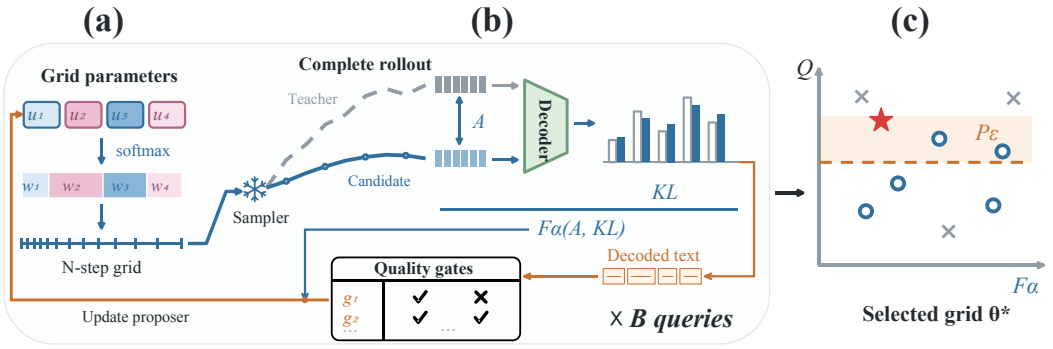


Figure 6: **Overview of StepTune.** Compress the schedule space, search using F_α under quality constraints, and select and freeze one schedule for reuse.

are then frozen and evaluated on ELF-B, ELF-M, and ELF-L, using 160 independent validation trajectories for each model and budget.

Figure 5 shows that the joint discrepancy decreases by approximately 68–80% relative to LogitNormal across the six validation settings. Crucially, both components decrease in every setting: latent error falls by 59.8–81.5%, and decoder KL by 71.6–78.2%. The improvement thus reflects reductions in both forms of deviation, rather than a decrease in one component offsetting an increase in the other.

Calibrating timestep placement with the joint objective substantially reduces both latent error and decoder KL, with these benefits persisting on unseen trajectories and across model scales.

4 StepTune: Tuning Timesteps for Continuous DLMs

In this section, we introduce **StepTune**, an algorithm that searches for improved timestep placements under a limited calibration budget, guided by the objective F_α in Equation (4). Each timestep placement defines a sampling schedule, an ordered sequence of time points used during generation. The algorithm proceeds in three stages (Figure 6): (I) *Compress* the schedule space, (II) *Search* using F_α under quality constraints, and (III) *Select and freeze* one schedule for reuse across inputs.

Stage I: Compress the schedule space. Let N denote the fixed NFE budget. Directly optimizing all $N-1$ interior timesteps creates a high-dimensional search problem. We therefore compress the schedule space to make the search tractable under a limited calibration budget.

We divide the sampling interval into R regions with boundaries $0 = b_0 < b_1 < \cdots < b_R = 1$. Each region r receives $n_r \geq 1$ uniformly spaced steps, with $\sum_{r=1}^{R} n_r = N$. We search only the interior boundaries and step allocations; these determine all timesteps while keeping the endpoints and total NFE fixed. Section 5.2 evaluates the query efficiency of this compressed search space.

Alternatively, we can restrict the search to a finite bank of schedules generated from chosen placement rules. Both approaches reduce the search space without changing the inference budget.

Stage II: Search under quality constraints. We use a calibration set of M instances, each consisting of an input and an initial noise realization. For each instance, we cache a high-NFE reference generated with the same input and initial noise. Each candidate schedule is evaluated on the entire set and scored using F_α, with the corresponding language quality metrics also recorded.

Quality constraints limit degradation in task performance and guard against undesirable generation behavior, such as excessive repetition or abnormal output length. For each requirement j, we define a margin $g_j(\theta)$ for schedule θ, with $g_j(\theta) \geq 0$ indicating satisfaction. The metrics and thresholds used in each experiment are detailed in Appendix B.

Let B denote the budget of schedule evaluations. Each evaluation runs one schedule over the entire calibration set. For the parameterized search, constrained Bayesian optimization proposes schedules

that balance predicted reductions in F_α with exploration. Predicted constraint violations penalize or exclude proposals, discouraging expenditure on schedules likely to violate the quality requirements. A finite candidate bank is evaluated directly. In both cases, complete sampling runs provide the measured discrepancy and quality margins used for final selection.

Stage III: Select and freeze. After search, let $\mathcal{H}_{\text{feas}}$ contain the evaluated schedules that satisfy all measured quality constraints. If this set is empty, we retain the baseline schedule. Otherwise, let $Q(\theta)$ denote the observed calibration quality, oriented so that larger values are better. We first retain schedules with near-best quality and then select the one with the lowest F_α.

For a scalar quality criterion, let $\varepsilon_Q \geq 0$ be a tolerance specified before search. The best observed feasible quality is $Q_{\max} = \max_{\theta \in \mathcal{H}_{\text{feas}}} Q(\theta)$. We define the retained set $\mathcal{P}_\varepsilon$ and selected schedule $\theta^\star$ as

$$\begin{aligned} \mathcal{P}_\varepsilon &= \{\theta \in \mathcal{H}_{\text{feas}} : Q(\theta) \geq Q_{\max} - \varepsilon_Q\}, \\ \theta^\star &= \arg\min_{\theta \in \mathcal{P}_\varepsilon} F_\alpha(\theta). \end{aligned} \tag{5}$$

For multiple quality metrics, the retained set uses the metric-wise or joint tolerance tests described in Appendix B. Quality therefore determines the admissible set, and F_α determines the final choice within it. The design of tolerance helps limit overfitting to calibration scores.

We freeze the selected timestep placement before evaluation on untouched test data and reuse it across inputs. Deployment changes only timestep placement, requiring no model updates, per-example search, backward passes, or additional model evaluations.

5 Experiments

Experimental setup. We evaluate StepTune with ELF (Hu et al., 2026) and AURORA-LM (Liang et al., 2026) on unconditional and conditional generation. For ELF, unconditional experiments use ELF-B at NFE–8/16/32 and report perplexity and unigram entropy. Conditional experiments use task-specific checkpoints on XSum and WMT14 De→En at NFE–16/32/64, reporting ROUGE-1/2/L and BLEU/chrF. For AURORA-LM-S, we evaluate unconditional generation and XSum summarization at NFE–16 using the same metrics, additionally reporting MAUVE for unconditional generation.

We compare against ELF's native LogitNormal schedule, AURORA-LM's native tan-d schedule, and an adaptation of LD3 for ELF conditional generation. Ablation and query-efficiency studies use the ELF WMT14 De→En checkpoint. At matched NFE, only timestep placement changes; the model and other sampling settings remain fixed. Schedules are calibrated on small, disjoint sets and frozen before held-out evaluation. Complete calibration, reference, sampling, evaluation, and efficiency protocols are provided in Appendix B.

5.1 Main Results

Performance on conditional generation. StepTune consistently improves conditional generation quality across tasks and sampling budgets. On ELF, it outperforms both native LogitNormal and the LD3 adaptation on every reported metric at NFE–16/32/64 (Table 1). At NFE–16, ROUGE-2 on XSum increases from 11.010 to 12.589, while BLEU on WMT14 increases from 23.030 to 26.839 relative to native sampling. By contrast, the LD3 adaptation improves XSum scores but falls below native sampling on both WMT14 metrics at NFE–32, despite using the same teacher budget.

More importantly, our NFE–16 schedules surpass native NFE–64 on all three ROUGE metrics and both BLEU and chrF. This quality advantage translates into measured efficiency: on XSum and WMT14, respectively, generation is $3.82\times$ and $3.84\times$ faster at batch size 1, with $3.74\times$ and $3.68\times$ the throughput at batch size 64. At matched NFE–16 and 32, latency and throughput remain essentially unchanged. Thus, timestep placement improves the quality–efficiency trade-off by enabling better generation with only one quarter of the native baseline's evaluations on both tasks.

The benefit also extends to AURORA-LM-S (Table 2b). On XSum, StepTune at NFE–16 exceeds native tan-d sampling at NFE–32 on all three ROUGE metrics, achieving 36.752/13.635/29.081 versus 36.580/13.479/28.873. Improved placement therefore yields higher quality with half the evaluations in a second continuous DLM architecture, despite differences in the latent representation, decoder, and native timestep schedule.

Table 1: **Conditional generation quality and efficiency.** Results use the full XSum and WMT14 test sets (11,334 and 3,003 examples). Lat. denotes latency in ms at batch size 1, and Thr. denotes throughput in sequences/s at batch size 64; both are measured on an NVIDIA B200. All baselines are reproduced. LD3 refers to our adaptation of the original method (Appendix B.5); LD3 and ours use a 64-NFE teacher for NFE–16/32 and a 128-NFE teacher for NFE–64.

NFE	Method	XSum					WMT14 De→En			
		R-1 ↑	R-2 ↑	R-L ↑	Lat. ↓	Thr. ↑	BLEU ↑	chrF ↑	Lat. ↓	Thr. ↑
16	Native LogitNormal	34.636	11.010	26.351	142.11	25.83	23.0295	51.0300	141.09	190.97
	LD3 adaptation	35.112	11.600	26.940	–	–	24.7199	52.2235	–	–
	Ours	**36.304**	**12.589**	**28.015**	142.20	25.83	**26.8390**	**53.6881**	141.07	191.00
32	Native LogitNormal	35.340	11.611	26.996	275.67	13.51	24.6668	52.1351	274.66	100.96
	LD3 adaptation	35.326	11.788	27.131	–	–	24.3254	51.8025	–	–
	Ours	**36.577**	**12.947**	**28.263**	275.75	13.51	**27.0917**	**53.7921**	274.46	101.00
64	Native LogitNormal	35.726	11.878	27.276	543.15	6.91	25.5353	52.7495	541.11	51.96
	LD3 adaptation	35.810	12.111	27.547	–	–	25.9150	53.0351	–	–
	Ours	**36.607**	**12.988**	**28.277**	543.19	6.91	**27.1268**	**53.8529**	541.03	51.99

Table 2: **Generation results on ELF-B and AURORA-LM-S.** (a) Unconditional generation on ELF-B, averaged over six runs. (b) XSum summarization and OpenWebText unconditional generation on AURORA-LM-S, averaged over five runs. All baselines are reproduced.

NFE	8	16	32
PPL ↓			
LogitNormal	69.169	32.563	24.021
Ours	**41.260**	**26.059**	**20.702**
Entropy			
LogitNormal	5.229	5.173	5.157
Ours	5.233	5.112	5.111

(a) ELF-B

Method	NFE	Metrics		
XSum		R-1 ↑	R-2 ↑	R-L ↑
Native tan-d	32	36.580	13.479	28.873
Ours	16	**36.752**	**13.635**	**29.081**
OpenWebText		PPL ↓	Entropy	MAUVE ↑
Native tan-d	16	56.394	5.386	0.9132
Ours	16	**41.210**	5.356	**0.9326**

(b) AURORA-LM-S

Performance on unconditional generation. The improvements also hold for unconditional generation. On ELF-B, StepTune reduces perplexity relative to native LogitNormal by 40.3%, 20.0%, and 13.8% at NFE 8, 16, and 32, respectively (Table 2a). The largest gain occurs at the smallest budget, where perplexity falls from 69.169 to 41.260, highlighting the importance of timestep placement under tight budgets. Across these budgets, unigram entropy remains within 1.2% of the native values, showing that the perplexity reductions are accompanied by little change in this diversity measure.

AURORA-LM-S exhibits the same benefit at NFE–16 (Table 2b). Perplexity decreases from 56.394 to 41.210, a 26.9% reduction, while MAUVE improves from 0.9132 to 0.9326 and entropy remains close to the native value (5.356 versus 5.386). The improvement thus extends beyond perplexity to distributional similarity with human text. Together, these results show that timestep calibration improves generation quality across architectures and task types.

5.2 Calibration-query efficiency

We examine how search-space compression affects search efficiency using the ELF WMT14 De→En checkpoint at NFE–16. We compare three search spaces under the same Bayesian optimization settings and a budget of 240 queries. After each query, we select the schedule with the lowest F_α on the calibration set so far and evaluate it on the validation set, plotting its measured validation F_α.

Region3 and Mixed6 both divide the sampling interval into four regions. Region3 fixes the step allocation to $(4, 4, 4, 4)$ and searches only the boundaries, whereas Mixed6 also varies the allocations, giving three and six degrees of freedom, respectively (Section 4). Full15 searches all 15 interior

timesteps without compression. Each search begins with one uniform schedule and 15 Sobol points, followed by Bayesian optimization. The shaded region in Figure 7 marks this initialization phase.

Mixed6 reaches validation $F_\alpha \leq 0.15$ after only 20 queries, compared with 63 for Region3 and 153 for Full15, and achieves the lowest final value within the 240-query budget. Region3 also outperforms Full15, showing that boundary compression alone improves search efficiency. Mixed6's further gains highlight the value of adapting step allocations alongside boundaries. Together, these results show that greater search-space flexibility does not necessarily yield better schedules under a limited query budget.

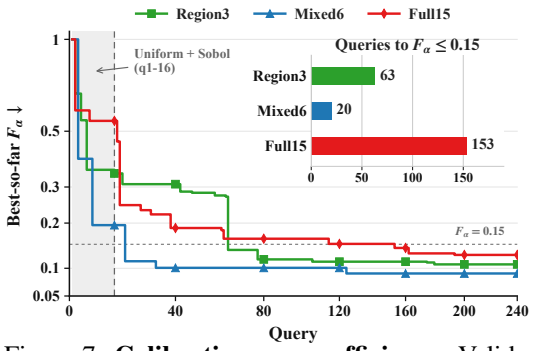


Figure 7: **Calibration-query efficiency.** Validation F_α of the best calibration schedule so far. Both axes use logarithmic compression.

We also report the actual search time used to obtain the ELF NFE–16 schedules evaluated in Table 1. These schedules were calibrated with 160 queries, taking 0.57 GPU-minutes for WMT14 and 6.79 GPU-minutes for XSum on a single NVIDIA B200. This one-time cost is amortized over subsequent generation by reusing the frozen schedules. Timing details are provided in Appendix B.

5.3 Ablation Study

We compare calibration objectives on WMT14 at NFE–16 using Region3 from Section 5.2 and ten independent calibration sets of 16 examples each. Table 3 reports test-set quality for the resulting frozen schedules, including variation across sets.

Table 3: **Objective ablation.** WMT14, NFE–16; mean ± standard deviation across ten calibration panels.

Method	BLEU ↑	chrF ↑
A-only	24.361 ± 0.153	52.653 ± 0.108
KL-only	24.357 ± 0.209	52.648 ± 0.142
Pure-F	24.515 ± 0.121	52.771 ± 0.067
Pure-Q	23.814 ± 1.975	52.227 ± 1.543
Full	$\mathbf{24.721} \pm 0.383$	$\mathbf{52.918} \pm 0.247$

A-only and KL-only both perform well (see Appendix B.5 for detailed definitions), with A-only achieving slightly higher mean quality and lower variation. This is consistent with latent error providing direct guidance at larger deviations, where it strongly correlates with KL. Pure-F improves both mean quality and stability, supporting the design rationale for the joint objective in Section 3.3.

Pure-Q yields lower mean quality and much larger variation. Discrete changes in decoded text make task scores an irregular search signal, while even high calibration scores can reflect overfitting to the small calibration set when F_α is ignored. Full combines F_α-guided search with measured quality and achieves the highest mean BLEU and chrF. Its variation is higher than Pure-F but substantially lower than Pure-Q, trading some stability for better generation quality—the ultimate goal of calibration.

6 Conclusion

The practical future of continuous DLMs depends on realizing the efficiency of parallel generation with few sequential denoising steps. Our work explores an opportunity shared by these models: timestep placement remains adjustable after training, offering a widely applicable, low-cost route to improving generation without updating model weights. Our empirical study demonstrates substantial and persistent effects on terminal representations and decoded token distributions, and identifies language-specific properties of the relationship between them.

Building on these findings, StepTune uses a language-aware objective and search-space compression to calibrate reusable schedules under generation-quality constraints. Experiments across architectures and tasks demonstrate consistent quality improvements that enable lower latency and higher throughput. The resulting schedules are reused across inputs, amortizing the one-time calibration cost over subsequent generation. More broadly, this work highlights timestep placement as a practical source of improvement in existing models, showing how better use of a limited sampling budget can help realize the efficiency gains promised by parallel language generation.

AI USE STATEMENT

In the preparation of this work, generative AI tools were employed solely as auxiliary aids. Specifically, they assisted with implementing and debugging parts of the codebase by translating algorithmic procedures and design specifications conceived by the authors into working code, and with refining the manuscript's language, including improving fluency, correcting grammatical and typographical errors, and fixing formatting inconsistencies. All scientific contributions of this paper, including the research motivation, the proposed method and its formulation, the theoretical derivations, the experimental protocol, and the interpretation of results, originate entirely from the authors. The AI tools were not involved in conceiving the research direction, designing the methodology or experiments, composing any substantive portion of the manuscript, producing or altering experimental data, establishing theoretical claims, or drawing scientific conclusions. All AI-assisted code and text were carefully inspected, validated, and where necessary revised by the authors prior to inclusion. The authors bear full responsibility for the accuracy and integrity of the published work.

ETHICS STATEMENT

StepTune improves the quality–efficiency trade-off of continuous diffusion language models by optimizing inference timestep placement while keeping model parameters fixed. Our experiments use existing model checkpoints and established benchmarks, including XSum, WMT14, and OpenWebText, with automated evaluation metrics. Reducing the number of denoising steps can lower inference costs and make language generation more accessible, but may also lower the cost of generating harmful or misleading content. StepTune does not specifically address biases, factual inaccuracies, or privacy risks associated with the underlying models and their training data. Its calibration constraints assess generation quality and do not constitute safety guarantees. Moreover, because timestep placement changes output distributions, keeping model weights fixed does not ensure that safety-related behavior remains unchanged. Deployment should therefore include application-specific safety evaluation, appropriate safeguards against misuse, and adherence to the terms of use of the underlying models and datasets.

REPRODUCIBILITY STATEMENT

The experimental settings needed to reproduce all reported results—checkpoint and sampler configurations, calibration and selection protocols, search budgets, and evaluation procedures—are specified in Appendix B. Every deployed timestep schedule is released in the accompanying reproducibility package as exact float32 grids, together with the compact coordinates and the deterministic constructor that regenerates them; these frozen operating points can be evaluated directly from the released checkpoints without re-running calibration.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021. doi: 10.48550/arXiv.2107.03006. URL https://arxiv.org/abs/2107.03006.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ales Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *ACL Workshop on Statistical Machine Translation*, 2014. doi: 10.3115/v1/w14-3302. URL https://doi.org/10.3115/v1/w14-3302.

Defang Chen, Zhenyu Zhou, Can Wang, Chunhua Shen, and Siwei Lyu. On the trajectory regularity of ODE-based diffusion sampling. In *ICML*, 2024. doi: 10.48550/arXiv.2405.11326. URL https://proceedings.mlr.press/v235/chen24bm.html.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. LangFlow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026. doi: 10.48550/arXiv.2604.11748. URL https://arxiv.org/abs/2604.11748.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022. doi: 10.48550/arXiv.2211.15089. URL https://arxiv.org/abs/2211.15089.

Eric Frankel, Sitan Chen, Jerry Li, Pang Wei Koh, Lillian J. Ratliff, and Sewoong Oh. S4S: Solving for a fast diffusion model solver. In *ICML*, 2025. doi: 10.48550/arXiv.2502.17423. URL https://proceedings.mlr.press/v267/frankel25a.html.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. In *NeurIPS*, 2023. doi: 10.52202/075280-0730. URL https://doi.org/10.52202/075280-0730.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026. doi: 10.48550/arXiv.2605.10938. URL https://arxiv.org/abs/2605.10938.

Yilie Huang, Wenpin Tang, and Xun Yu Zhou. ART for diffusion sampling: Continuous-time control and actor-critic learning. *arXiv preprint arXiv:2607.02137*, 2026. doi: 10.48550/arXiv.2607.02137. URL https://arxiv.org/abs/2607.02137.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *NeurIPS*, 2022. doi: 10.48550/arXiv.2205.14217. URL https://arxiv.org/abs/2205.14217.

Jiajun Liang, Yucheng Liao, Yukang Cao, Jiazhe Wei, Ken Li, Wende Tan, Jiankun Zhang, ZY Cui, Jingkang Yang, Liucheng Guo, Shiqi Yang, B. Yang, Caifeng Shan, Ziwei Liu, and Chenyang Si. AURORA-LM: Autoencoding unified representation for continuous-latent diffusion language modeling. *arXiv preprint arXiv:2608.02602*, 2026. doi: 10.48550/arXiv.2608.02602. URL https://arxiv.org/abs/2608.02602.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *EMNLP*, 2018. doi: 10.18653/v1/D18-1206. URL https://aclanthology.org/D18-1206/.

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. In *NeurIPS*, 2025. doi: 10.52202/085713-1689. URL https://doi.org/10.52202/085713-1689.

Jianning Pei, Han Hu, and Shuyang Gu. Optimal stepsize for diffusion sampling. *arXiv preprint arXiv:2503.21774*, 2025. doi: 10.48550/arXiv.2503.21774. URL https://arxiv.org/abs/2503.21774.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. In *ICML*, 2024. doi: 10.48550/arXiv.2404.14507. URL https://arxiv.org/abs/2404.14507.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *NeurIPS*, 2024. doi: 10.52202/079017-4135. URL https://doi.org/10.52202/079017-4135.

Neta Shaul, Uriel Singer, Ricky T. Q. Chen, Matthew Le, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke non-stationary solvers for fast sampling of diffusion and flow models. In *ICML*, 2024. doi: 10.48550/arXiv.2403.01329. URL https://arxiv.org/abs/2403.01329.

Vinh Tong, Trung-Dung Hoang, Anji Liu, Guy Van den Broeck, and Mathias Niepert. Learning to discretize denoising diffusion ODEs. In *ICLR*, 2025. doi: 10.48550/arXiv.2405.15506. URL https://arxiv.org/abs/2405.15506.

Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *ICLR*, 2022. doi: 10.48550/arXiv.2202.05830. URL https://arxiv.org/abs/2202.05830.

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. Continuous diffusion scales competitively with discrete diffusion for language. *arXiv preprint arXiv:2605.18530*, 2026. doi: 10.48550/arXiv.2605.18530. URL https://arxiv.org/abs/2605.18530.

Liangyu Yuan, Ruoyu Wang, Tong Zhao, Dingwen Fu, Mingkun Lei, Beier Zhu, and Chi Zhang. Few-step diffusion sampling through instance-aware discretizations. In *CVPR*, 2026. doi: 10.48550/arXiv.2603.17671. URL https://arxiv.org/abs/2603.17671.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. doi: 10.1109/cvpr.2018.00068. URL https://doi.org/10.1109/cvpr.2018.00068.

Travis Zhang, Christian Belardi, Justin Lovelace, Jin Peng Zhou, Saebyeol Shin, Carla P. Gomes, and Kilian Q. Weinberger. Optimize your sampling: Tuned diffusion sampling with bayesian optimization. *arXiv preprint arXiv:2608.18040*, 2026. doi: 10.48550/arXiv.2608.18040. URL https://arxiv.org/abs/2608.18040.

Aihua Zhu, Rui Su, Qinglin Zhao, Li Feng, Meng Shen, and Shibo He. Hierarchical schedule optimization for fast and robust diffusion model sampling. In *AAAI*, 2026. doi: 10.1609/aaai.v40i16.38400. URL https://doi.org/10.1609/aaai.v40i16.38400.

APPENDIX

A THEORETICAL ANALYSIS

This appendix makes precise how timestep placement affects the terminal latent state and its decoded distribution. The argument has four parts. First, we write the self-conditioned sampler as a map on an augmented state and derive an exact, pathwise decomposition of its terminal error. Second, we separate this exact identity from the reference-Jacobian approximation and bound the approximation remainder. Third, we connect terminal latent error to decoder KL divergence, both locally and at a finite displacement. Finally, we show which quantities must remain stable for one fixed schedule to retain low teacher-relative error across inputs and random seeds.

Throughout, $\|\cdot\|_2$ denotes the Euclidean norm and $\|\cdot\|_{\mathrm{op}}$ its induced operator norm. All statements are for a fixed model, sampler, number of function evaluations, and generation setting. Random samplers are treated pathwise after their random driver has been fixed.

A.1 SAMPLER, SCHEDULE, AND NUMERICAL REFERENCE

Let

$$\theta = (t_0, \dots, t_N), \qquad 0 = t_0 < t_1 < \cdots < t_N = 1, \qquad h_k = t_{k+1} - t_k,$$

be an N-step schedule. A self-conditioned sampler carries both a latent state z_k and a memory state s_k, such as the model's previous endpoint prediction. We therefore use the augmented state

$$y_k = (z_k, s_k) \in \mathcal{Y}.$$

For example, a deterministic Euler step can be written abstractly as

$$p_k = g_x(z_k, s_k, t_k), \qquad z_{k+1} = z_k + h_k v_x(z_k, s_k, t_k), \qquad s_{k+1} = p_k,$$

where g_x includes the conditioning and guidance operations and v_x includes the parameterization-specific conversion from the model prediction to a velocity. The analysis does not depend on this particular parameterization. It only requires the complete deployed update to define a map

$$y_{k+1} = \Phi^{\theta}_{x,k}(y_k; \omega), \tag{6}$$

where ω collects any random variables used by the rollout. Crucially, $\Phi^{\theta}_{x,k}$ includes both the latent update and the memory refresh. Omitting the latter would remove a genuine route by which an early error affects later model evaluations.

Let y^*_k denote a reference state at the same boundary t_k. In our application these states come from a refined numerical rollout with matched conditioning, initialization, and sampler randomness. The reference need not be the exact solution of a continuous-time process. Define the boundary error and one-step defect by

$$e_k := y_k - y^*_k, \qquad d^{\theta}_k := y^*_{k+1} - \Phi^{\theta}_{x,k}(y^*_k; \omega). \tag{7}$$

These quantities answer different questions. The error e_k records the discrepancy already present at boundary k. The defect d^{θ}_k starts the coarse update from the reference state and measures the new discrepancy created on interval $[t_k, t_{k+1}]$. This distinction is what permits interval-level attribution.

A.2 EXACT TRANSPORT OF INTERVAL DEFECTS

We first state the result without a small-error approximation.

Theorem 1 (Exact pathwise defect transport)**.** *Fix x, θ, and the realized driver ω. Suppose that $\Phi^{\theta}_{x,k}$ is continuously differentiable on a neighborhood of the line segment joining y^*_k and y_k, for every k. Define the secant operator*

$$\bar{A}^{\theta}_k := \int_0^1 D_y \Phi^{\theta}_{x,k}(y^*_k + \tau e_k; \omega)\, d\tau \tag{8}$$

and, for $0 \le j < N$,

$$\bar{P}^{\theta}_{j:N} := \bar{A}^{\theta}_{N-1} \bar{A}^{\theta}_{N-2} \cdots \bar{A}^{\theta}_{j}, \qquad \bar{P}^{\theta}_{N:N} := I. \tag{9}$$

Then

$$e_N = \bar{P}^{\theta}_{0:N} e_0 - \sum_{k=0}^{N-1} \bar{P}^{\theta}_{k+1:N} d^{\theta}_k. \tag{10}$$

If the candidate and reference share their augmented initialization, then $e_0 = 0$ *and*

$$e_N = -\sum_{k=0}^{N-1} \bar{P}^{\theta}_{k+1:N} d^{\theta}_k. \tag{11}$$

Proof. Subtract the reference state at the next boundary and insert the defect from Equation (7):

$$e_{k+1} = \Phi^{\theta}_{x,k}(y_k;\omega) - y^*_{k+1} \tag{12}$$

$$= \Phi^{\theta}_{x,k}(y_k;\omega) - \Phi^{\theta}_{x,k}(y^*_k;\omega) - d^{\theta}_k. \tag{13}$$

The fundamental theorem of calculus along the segment from y^*_k to y_k gives

$$\Phi^{\theta}_{x,k}(y_k;\omega) - \Phi^{\theta}_{x,k}(y^*_k;\omega) = \bar{A}^{\theta}_k e_k.$$

Hence

$$e_{k+1} = \bar{A}^{\theta}_k e_k - d^{\theta}_k. \tag{14}$$

Repeated substitution proves Equation (10); setting $e_0 = 0$ yields Equation (11). □

The theorem separates two roles of the schedule. The defect d^{θ}_k measures where the coarse grid creates error, while $\bar{P}^{\theta}_{k+1:N}$ describes how all subsequent updates carry that error to the endpoint. An early defect passes through more later operators, but this does not by itself imply amplification: those operators may amplify, attenuate, rotate, or cancel its contribution. The terminal discrepancy therefore depends on both the magnitudes and directions of the transported defects.

Equation (11) is exact at the deployed NFE. It requires neither independent defects, invertible steps, nor small intermediate errors. Its pathwise form also allows the transport operators and defects to be statistically dependent. Timestep placement influences both objects, because changing a boundary changes the one-step approximation as well as the states at which later updates are evaluated.

A.3 Reference linearization and its remainder

The secant operators are exact, but they depend on the complete candidate–reference displacement. A simpler approximation replaces each secant operator with the Jacobian at the reference state. The next result makes the resulting error explicit.

Proposition 2 (Reference-Jacobian approximation)**.** *Assume the conditions of Theorem 1. Suppose additionally that each realized step map is twice continuously differentiable on the relevant state segment and that*

$$\|D_y \Phi^{\theta}_{x,k}(y;\omega)\|_{\mathrm{op}} \le L_k, \qquad \|D^2_y \Phi^{\theta}_{x,k}(y;\omega)\|_{\mathrm{op}} \le B_k$$

there. Let

$$A^{\theta}_k := D_y \Phi^{\theta}_{x,k}(y^*_k;\omega), \qquad P^{\theta}_{j:N} := A^{\theta}_{N-1} \cdots A^{\theta}_j, \qquad P^{\theta}_{N:N} := I.$$

For matched initialization,

$$e_N = -\sum_{k=0}^{N-1} P^{\theta}_{k+1:N} d^{\theta}_k + R_N, \tag{15}$$

where

$$\|R_N\|_2 \le \frac{1}{2} \sum_{k=0}^{N-1} \left(\prod_{j=k+1}^{N-1} L_j \right) B_k \|e_k\|_2^2. \tag{16}$$

An empty product in Equation (16) equals one.

Proof. Taylor's theorem at y_k^* gives

$$e_{k+1} = A_k^\theta e_k - d_k^\theta + r_k, \qquad \|r_k\|_2 \le \frac{1}{2} B_k \|e_k\|_2^2. \tag{17}$$

Unrolling Equation (17) with $e_0 = 0$ yields

$$R_N = \sum_{k=0}^{N-1} P_{k+1:N}^\theta r_k.$$

The claimed bound follows from the triangle inequality, $\|P_{k+1:N}^\theta\|_{\mathrm{op}} \le \prod_{j=k+1}^{N-1} L_j$, and the bound on r_k. □

Equation (16) shows exactly when the approximation can fail: curvature at step k is multiplied by the squared error already present at that step and then transported through the remaining dynamics. Thus the reference-Jacobian approximation is a local statement. At a large student–reference gap, the exact secant identity remains valid but the quadratic remainders need not be small.

A.4 Random samplers and coarse–fine coupling

State derivatives of a realized update. Fixing the random driver leaves an ordinary map of the augmented state. For example, an Euler–Maruyama latent update may take the form

$$z_{k+1} = z_k + h_k b_k(y_k) + \sum_m \sigma_{k,m}(y_k)\, \Delta W_{k,m}. \tag{18}$$

Conditional on the realized increments, its state derivative contains

$$I + h_k \partial_z b_k + \sum_m \Delta W_{k,m}\, \partial_z \sigma_{k,m}. \tag{19}$$

The derivatives with respect to the memory state and the derivatives of the memory refresh are included in the augmented Jacobian. We differentiate the realized finite-step map with respect to its state; we do not differentiate a Brownian path with respect to time. The smoothness assumption of Theorem 1 applies to this complete realized map.

Coupling across grids. For a genuine SDE discretization, a common Brownian path can be constructed on the union grid $\{\tau_j\}$ of the candidate and reference boundaries. Draw independent $\eta_j \sim \mathcal{N}(0, I)$ and set

$$\Delta W_j = \sqrt{\tau_{j+1} - \tau_j}\, \eta_j, \qquad \Delta W_k^\theta = \sum_{j:\,[\tau_j, \tau_{j+1}] \subseteq [t_k, t_{k+1}]} \Delta W_j. \tag{20}$$

Summing the same fine increments over each coarse interval preserves the correct marginal law on both grids and produces a common-path coupling.[1] Additional grid points can be introduced using Brownian bridges conditional on the already fixed path values.

For a discrete Gaussian-reinjection sampler, the coupling instead specifies how a shared collection of standard Gaussian draws is assigned to the two rollouts while preserving each rollout's marginal law. Matching draws by their index is one valid choice. Reinjection amplitudes and shifted evaluation times remain part of Φ_k^θ. Such a coupling need not represent one Brownian path: the transport theorem concerns the deployed finite-step maps and does not require an underlying SDE convergence argument.

Numerical references. The reference states in the transport identity may themselves be numerical. A refinement containing the candidate boundaries, or an interpolation supplying both latent and memory components, defines the required states. The resulting reference discrepancy is absorbed into d_k^θ. When connecting this decomposition to F_α, the reference endpoint must be the same Z_x^* used to evaluate that objective. The algebra is valid for any such reference under a fixed coupling; it does not identify a numerical teacher with the exact solution of a continuous-time process.

[1] M. B. Giles, Multilevel Monte Carlo Path Simulation, *Operations Research*, 56(3):607–617, 2008.

A.5 From terminal latent error to decoder KL

Theorem 3 (Local decoder pullback). *Let Z_x^* and $Z_x(\theta)$ be the reference and candidate terminal latent states for example x. At retained token position i, let $\ell_{i,x}(Z) \in \mathbb{R}^V$ be the decoder logits, $q_{i,x}(Z) = \mathrm{softmax}(\ell_{i,x}(Z))$, and $q_{i,x}^* := q_{i,x}(Z_x^*)$. With fixed weights $w_i \geq 0$ satisfying $\sum_i w_i = 1$, define the forward teacher-to-candidate KL*

$$K_x(Z) := \sum_i w_i \,\mathrm{KL}\big(q_{i,x}^* \,\big\|\, q_{i,x}(Z)\big)\,. \tag{21}$$

If every $\ell_{i,x}$ is differentiable at Z_x^*, then, as $e \to 0$,

$$K_x(Z_x^* + e) = \frac{1}{2} e^\top W_{\mathrm{dec},x} e + o(\|e\|_2^2), \tag{22}$$

where

$$W_{\mathrm{dec},x} := \sum_i w_i J_{i,x}^\top F(q_{i,x}^*) J_{i,x}, \qquad J_{i,x} := D_Z \ell_{i,x}(Z_x^*), \tag{23}$$

and

$$F(q) := \mathrm{diag}(q) - qq^\top.$$

In particular, $W_{\mathrm{dec},x}$ is positive semidefinite.

Proof. For one position, let $a^* = \ell_{i,x}(Z_x^*)$, $\delta a = \ell_{i,x}(Z_x^* + e) - a^*$, and $\psi(a) = \log \sum_v \exp(a_v)$. Direct substitution gives

$$\mathrm{KL}(q_{i,x}^* \| q_{i,x}(Z_x^* + e)) = \psi(a^* + \delta a) - \psi(a^*) - (q_{i,x}^*)^\top \delta a. \tag{24}$$

Because $\nabla\psi(a^*) = q_{i,x}^*$ and $\nabla^2\psi(a^*) = F(q_{i,x}^*)$, Taylor expansion cancels the linear term. Using $\delta a = J_{i,x} e + o(\|e\|_2)$, then summing over positions, proves Equation (22). Positive semidefiniteness follows from

$$b^\top F(q) b = \sum_v q_v b_v^2 - \left(\sum_v q_v b_v\right)^2 = \mathrm{Var}_{v\sim q}[b_v] \geq 0. \tag{25}$$

For sequence outputs, summing with the fixed token weights $w_i \geq 0$ gives the stated $W_{\mathrm{dec},x}$. □

The pullback in Equation (23) identifies latent directions to which the decoder is locally sensitive. It is not a finite-distance identity. The following two results describe the measured KL away from the reference without replacing it by the local Fisher quadratic.

Proposition 4 (Finite logit-displacement bound). *For natural logarithms and arbitrary candidate logits,*

$$K_x(Z) \leq \frac{1}{4} \sum_i w_i \|\ell_{i,x}(Z) - \ell_{i,x}(Z_x^*)\|_2^2. \tag{26}$$

Proof. For one position, Taylor's integral formula applied to ψ gives

$$\mathrm{KL}(q^* \| q) = \int_0^1 (1-\tau)\, \delta a^\top F(\mathrm{softmax}(a^* + \tau\delta a)) \delta a \, d\tau.$$

The absolute row sums of $F(q)$ are $2q_v(1-q_v) \leq 1/2$, so $\|F(q)\|_{\mathrm{op}} \leq 1/2$. Integrating $1-\tau$ over $[0,1]$ gives the factor $1/4$. Summing with the fixed weights proves Equation (26). □

Proposition 5 (Exact decomposition along a finite latent path). *Let $e = Z_x(\theta) - Z_x^*$, $\gamma(\tau) = Z_x^* + \tau e$, and assume the decoder logits are twice continuously differentiable along γ. Write*

$$\ell_{i,\tau} := \ell_{i,x}(\gamma(\tau)), \qquad q_{i,\tau} := \mathrm{softmax}(\ell_{i,\tau}), \qquad q_i^* := q_{i,0},$$

and let dots denote derivatives with respect to τ. Define

$$g_\gamma(\tau) := \sum_i w_i \dot{\ell}_{i,\tau}^\top F(q_{i,\tau}) \dot{\ell}_{i,\tau}, \tag{27}$$

$$r_\gamma(\tau) := \sum_i w_i (q_{i,\tau} - q_i^*)^\top \ddot{\ell}_{i,\tau}. \tag{28}$$

Then

$$K_x(Z_x(\theta)) = G_\gamma + R_\gamma, \qquad G_\gamma := \int_0^1 (1-\tau) g_\gamma(\tau)\, d\tau, \qquad R_\gamma := \int_0^1 (1-\tau) r_\gamma(\tau)\, d\tau. \quad (29)$$

Here $G_\gamma \geq 0$, whereas R_γ can have either sign.

Proof. Let $K(\tau) := K_x(\gamma(\tau))$. Direct differentiation gives

$$K'(\tau) = \sum_i w_i (q_{i,\tau} - q_i^*)^\top \dot{\ell}_{i,\tau}$$

and therefore $K''(\tau) = g_\gamma(\tau) + r_\gamma(\tau)$. Since $K(0) = K'(0) = 0$, Taylor's integral formula yields Equation (29). The nonnegativity of G_γ follows from the positive semidefiniteness of $F(q)$. □

Theorem 3 describes the infinitesimal geometry at the reference. Proposition 5 is an exact identity for a finite latent displacement and exposes the additional curvature of the decoder logits along that path. Proposition 4 is a global upper bound in logit space. These statements are complementary and should not be used interchangeably.

A.6 Terminal normalization and the calibration objective

Let M_x select the terminal latent region on which fidelity is evaluated. For unconditional generation, it selects all generated latent positions. For conditional generation, it selects the generated suffix and excludes conditioning positions. With a fixed floor $\tau_A > 0$, define

$$A_x(\theta) := \frac{\|M_x(Z_x(\theta) - Z_x^*)\|_2^2}{D_{A,x}}, \qquad D_{A,x} := \max\{\|M_x Z_x^*\|_2^2, \tau_A\}. \quad (30)$$

For the decoder term, the retained positions are fixed by the teacher: we include positions through the teacher's first EOS token, inclusive, or the full generated region if no EOS appears. Retained positions have uniform weights summing to one. This teacher-defined mask prevents an early student termination from shortening its own evaluation range.

For a calibration set $\mathcal{C}$, let

$$\bar{A}(\theta) := \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} A_x(\theta), \qquad \bar{K}(\theta) := \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} K_x(Z_x(\theta)).$$

Given fixed positive scales A_{scale} and K_{scale}, StepTune uses

$$F_\alpha(\theta) := \alpha \frac{\bar{A}(\theta)}{A_{\text{scale}}} + (1-\alpha) \frac{\bar{K}(\theta)}{K_{\text{scale}}}, \qquad 0 \leq \alpha \leq 1. \quad (31)$$

The masks, denominator floors, and scales are held fixed across candidate schedules.

For one example, define

$$W_{A,x} := \frac{M_x^\top M_x}{D_{A,x}}, \qquad W_{\alpha,x} := \frac{2\alpha}{A_{\text{scale}}} W_{A,x} + \frac{1-\alpha}{K_{\text{scale}}} W_{\text{dec},x}.$$

Combining the per-example objective $F_{\alpha,x}(\theta) := \alpha\, A_x(\theta)/A_{\text{scale}} + (1-\alpha)\, K_x(Z_x(\theta))/K_{\text{scale}}$ with Theorem 3 yields the local form

$$F_{\alpha,x}(\theta) = \frac{1}{2} (e_x^Z)^\top W_{\alpha,x} e_x^Z + o(\|e_x^Z\|_2^2). \quad (32)$$

The latent term penalizes every selected latent direction, including directions to which the decoder is locally insensitive; the KL term weights directions by their effect on token probabilities.

Let Q_z project the augmented terminal error onto its latent component. Under matched initialization, Theorem 1 gives

$$e_x^Z = -\sum_{k=0}^{N-1} u_{k,x}^\theta, \qquad u_{k,x}^\theta := Q_z \bar{P}_{k+1:N}^\theta d_k^\theta.$$

Substitution into Equation (32) makes the interaction among intervals explicit:

$$F_{\alpha,x}(\theta) = \frac{1}{2}\sum_{k=0}^{N-1}\sum_{\ell=0}^{N-1}(u_{k,x}^{\theta})^{\top}W_{\alpha,x}u_{\ell,x}^{\theta} + o(\|e_x^Z\|_2^2). \tag{33}$$

The diagonal terms measure the contribution of individual transported defects; the off-diagonal terms record reinforcement or cancellation between intervals. StepTune evaluates the actual latent error and decoder KL in Equation (31). Equations (32)–(33) explain the geometry of those two measurements but do not replace them during search.

A.7 When a fixed schedule can transfer

A reusable schedule need not produce similar terminal states for different inputs. It must keep each candidate trajectory close to its own paired reference. We formalize this distinction for a fixed schedule θ.

Let $\xi = (x, z_0, \omega)$ denote the conditioning input, initial latent, and subsequent random driver. For two realizations ξ and ξ' with compatible state coordinates, define

$$\Delta e_k := e_k(\xi) - e_k(\xi'), \quad \Delta d_k := d_k(\xi) - d_k(\xi'), \quad \Delta \bar{A}_k := \bar{A}_k(\xi) - \bar{A}_k(\xi').$$

Proposition 6 (Cross-realization stability bound)**.** *Suppose the conditions of Theorem 1 hold for both realizations and each candidate is paired with a reference that shares its own augmented initialization. Then*

$$\|\Delta e_N\|_2 \le \sum_{k=0}^{N-1} \|\bar{P}_{k+1:N}(\xi)\|_{\mathrm{op}} \left(\|\Delta d_k\|_2 + \|\Delta \bar{A}_k\|_{\mathrm{op}}\|e_k(\xi')\|_2\right). \tag{34}$$

Proof. Subtract the two exact one-step recursions and add and subtract $\bar{A}_k(\xi)e_k(\xi')$:

$$\Delta e_{k+1} = \bar{A}_k(\xi)\Delta e_k + \Delta \bar{A}_k e_k(\xi') - \Delta d_k. \tag{35}$$

Although z_0 may differ across realizations, each candidate and its own reference share their initial state. Hence $e_0(\xi) = e_0(\xi') = 0$ and $\Delta e_0 = 0$. Unrolling Equation (35), followed by the triangle inequality and submultiplicativity, proves Equation (34). □

The bound identifies a sufficient mechanism for reuse. If the transported change in interval defects is controlled, the transport operators do not vary too sharply, and the paired intermediate errors remain small, then teacher-relative terminal errors can remain similar across otherwise different trajectories. Operator variation is multiplied by the paired error $e_k(\xi')$, not by the absolute reference state. This permits a fixed grid to address recurring numerical-error structure while the input and random seed continue to determine the generated content.

This is a conditional explanation, not a distribution-free guarantee. Transfer of the calibration objective additionally requires comparable normalization and decoder sensitivity across examples; transfer to language quality requires the quality constraints and held-out evaluations described in the main text.

B Experimental Details

This appendix gives the calibration and evaluation protocols for the reported generation results. Calibration data are disjoint from validation and test data. Each selected schedule is frozen before held-out evaluation and reused across inputs.

B.1 Models, precision, and controlled variables

ELF checkpoints. The cross-scale and unconditional experiments use the PyTorch ELF checkpoints trained on OpenWebText. The ELF-B, ELF-M, and ELF-L checkpoints are taken at training steps 95,085, 76,070, and 57,051, respectively. Conditional generation uses the task-specific ELF-B checkpoints for XSum at step 39,800 and WMT14 De→En at step 880,600, together with a frozen T5-small encoder. ELF generation uses the T5-small tokenizer. Standard generation rollouts use bfloat16 model computation and float32 timestep grids.

What changes across candidate schedules. Within each deployment setting, all candidates use the same frozen checkpoint, sampler update, guidance parameters, endpoint times, and number of model evaluations. The candidate variable is the placement of the interior timesteps. Candidate schedules evaluated on the same calibration example share its conditioning input and initial latent. Random samplers additionally use the fixed coupling specified for that experiment. Thus paired differences within a setting isolate timestep placement; the higher-NFE teacher is a separate reference configuration and may use a different NFE or stochastic coefficient, as stated below.

Calibration, selection, and evaluation. One calibration query evaluates one complete grid on every example in the calibration panel. Teacher terminal latents and token distributions are computed once and cached. Schedule search and all quality gates use only the calibration panel. The selected schedule is then frozen and evaluated on disjoint validation seeds or held-out task examples. If no candidate satisfies the stated feasibility requirements, the baseline schedule is retained.

We use $F = F_{1/2}$ from Equation (4). Its components are the calibration means of relative squared terminal-latent error and forward decoder KL, $\mathrm{KL}(q_{\mathrm{teacher}} \| q_{\mathrm{candidate}})$. The latent region, teacher-EOS mask, and per-example normalizations follow Appendix A. The two normalization constants are fixed before comparing candidates; their sources are specified below.

B.2 Schedule parameterization and search budget

Four-region schedules. Conditional generation and the cross-scale control experiment use four contiguous trajectory regions. A softmax maps four boundary logits $\beta = (\beta_1, \ldots, \beta_4)$, with the gauge $\beta_1 = 0$, to positive region widths and cumulative boundaries B_r. When allocations are optimized, three additional free logits determine a four-way allocation softmax: we first assign one interval to every region and distribute the remaining $N - 4$ intervals by largest-remainder rounding, giving positive integers n_r with $\sum_r n_r = N$. Region r then contains n_r equal-width intervals on $[B_{r-1}, B_r]$, and the union of their left endpoints, followed by the terminal endpoint 1, forms a grid with exactly N solver intervals.

Bayesian search for ELF. Unless a setting below fixes the allocation, the search has six free coordinates: three boundary coordinates in $[-3, 3]$ and three allocation coordinates in $[-3.5, 3.5]$. Every search begins with Uniform and 15 scrambled Sobol proposals. Later queries are issued in batches of four using a Gaussian-process lower-confidence-bound acquisition function with a Matérn-$5/2$ kernel, length scale 0.25, and exploration coefficient 1.5. Teacher fidelity is the acquisition objective. In task experiments, measured quality constraints determine feasibility and may also penalize predicted violations. The reported schedule is selected only from fully evaluated candidates.

Finite-bank search. ELF-B unconditional generation does not use the continuous parameterization. At each NFE it evaluates a fixed bank of 80 grids drawn from ELF's native LogitNormal grid family.

Table 4 summarizes the teacher and query budgets. A budget is reported per deployment NFE; each grid evaluation covers the entire calibration panel.

Table 4: **ELF calibration configurations.** Budgets are per NFE. Each grid evaluation covers the full calibration panel; ELF-B unconditional calibration evaluates a fixed bank.

Experiment	NFE	Panel size	Grid evaluations	Teacher NFE
Schedule control	16/32	16	160	64
Unconditional	8/16/32	64	80 (bank)	64
XSum	16/32	64	160	64
XSum	64	64	160	128
WMT14	16/32	16	160	64
WMT14	64	16	160	128

B.3 Cross-scale schedule-control experiment

This experiment asks whether a schedule calibrated on one ELF model scale retains low teacher-relative error on other scales. We evaluate ELF-B, ELF-M, and ELF-L at deployment NFE 16 and 32. All runs use sequence length $1{,}024$, Euler ODE sampling, classifier-free-guidance scale 1,

self-conditioning-guidance scale 3, and noise scale 2. The teacher uses the same checkpoint as its paired candidate and a uniform NFE-64 grid.

For each deployment NFE, the calibration panel contains 16 ELF-M seeds and the validation panel contains 160 disjoint seeds. We run one 160-query search on ELF-M and retain the observed grid with the lowest calibration F; no task-quality criterion is used. This produces exactly two schedules, one for NFE 16 and one for NFE 32. Each schedule is then applied unchanged to ELF-B, ELF-M, and ELF-L on the validation panel. No schedule is recalibrated for ELF-B or ELF-L.

For the latent term, terminal MSE is normalized by the paired teacher's terminal energy. The decoder term uses the forward KL through the teacher's first EOS token, inclusive. For reporting on a given model, NFE, and split, each component is divided by its mean over 32 native LogitNormal replays; the native mean of the resulting joint objective is therefore one. Search itself uses only the ELF-M calibration normalizers. Confidence intervals use 16,384 crossed-bootstrap draws: validation seeds and native replays are resampled independently, and the reporting normalizers are recomputed within each draw.

B.4 Unconditional generation on ELF-B

Sampling and calibration. We use ELF-B with sequence length $1{,}024$ and the official SDE-inspired sampler. Classifier-free-guidance and self-conditioning-guidance scales are 1 and 3, respectively. The stochastic coefficient is $\gamma = 2.0$ at deployment NFE 8 and 16 and $\gamma = 1.5$ at NFE 32; the final update is deterministic. The paired teacher uses a uniform NFE-64 grid and $\gamma = 1$.

For each deployment NFE, the calibration panel contains 64 seeds. We evaluate 80 native LogitNormal grids with parameters $(-1.5, 0.8)$. The bank means of latent error and decoder KL define A_{scale} and K_{scale}, respectively. A grid is feasible only if $F < 1$ and it satisfies all applicable requirements in Table 5. Among the feasible grids, let $\mathrm{PPL}_{\min}$ be the lowest calibration GPT-2-Large perplexity. We retain grids with perplexity at most $1.05\,\mathrm{PPL}_{\min}$, then select the retained grid with the lowest F. Exact ties are resolved first by lower perplexity and then by candidate identifier. One schedule is selected at each NFE.

Held-out evaluation. For every deployment NFE, the frozen StepTune schedule and the matched native baseline are each evaluated in six independent runs of 1,000 generations. At batch size one, the native baseline draws a fresh LogitNormal grid for every example, while StepTune reuses its one frozen grid. Both methods use the same checkpoint, sampler configuration, and NFE.

We report GPT-2-Large perplexity and unigram entropy under the GPT-2 tokenizer. We additionally record repeated-trigram rate, distinct bigrams, generated length, EOS rate, and empty-output rate. Paired uncertainty for perplexity is computed in token-negative-log-likelihood space and transformed back to the reported scale. The hierarchical bootstrap uses 16,384 replicates, resampling runs and then examples within each sampled run. The matched PyTorch baseline is the basis of the controlled comparison.

Table 5: **ELF calibration quality requirements.** Subscript 0 denotes the bank reference for unconditional generation and the uniform student schedule for conditional generation.

Quantity	Unconditional	Conditional
Unigram entropy H	$\geq 0.98H_0$	–
Relative length L/L_0	$[0.90, 1.10]$	$[0.80, 1.20]$
Repeated-trigram rate R_3	$\leq \max(1.10R_{3,0}, R_{3,0} + 0.01)$	$\leq \max(1.50R_{3,0}, R_{3,0} + 0.01)$
EOS rate	$\geq \mathrm{EOS}_0 - 0.05$	$\geq \mathrm{EOS}_0 - 0.05$
Empty-output rate	$\leq \mathrm{Empty}_0 + 0.01$	$\leq \mathrm{Empty}_0 + 0.01$

B.5 Conditional generation on ELF-B

Shared sampling configuration. XSum and WMT14 De→En keep the conditioning positions fixed and generate a 64-token suffix. Candidate rollouts use Euler ODE sampling, classifier-free-guidance scale 2, self-conditioning-guidance scale 1, and decoder scale 1. XSum uses 64 training pairs for calibration; WMT14 uses 16. The paired teacher is Uniform at NFE 64 for deployment NFE 16 and

32, and Uniform at NFE 128 for deployment NFE 64. For each task and NFE, the corresponding Uniform student's mean latent error and decoder KL define the two scales in Equation (4).

Search space and budget. All six conditional settings use Mixed6, optimizing both the three interior region boundaries and the step allocations. Each search evaluates 160 grids using the initialization and Gaussian-process configuration in Appendix B.2.

Feasibility. At calibration time, every XSum ROUGE-1/2/L score, or both WMT14 sentence BLEU and sentence chrF, must be no more than 0.5 points below Uniform. A candidate must also satisfy the task-applicable requirements in Table 5. Rather than accepting a constraint from its point estimate alone, we require at least 0.90 paired-bootstrap support for the joint feasibility event under 8,192 fixed bootstrap draws.

Final selection. Selection is performed only among feasible, fully evaluated candidates. For XSum, the scalar calibration-quality score is the equal-weight mean of Uniform-normalized ROUGE-1/2/L. For WMT14, it is the equal-weight mean of Uniform-normalized sentence BLEU, sentence chrF, corpus BLEU, and corpus chrF. Let the common best candidate maximize this score over the feasible observations. The XSum near-best set retains a candidate only if each of its three ROUGE metrics has bootstrap probability at least 0.90 of being no more than 0.5 points below the corresponding metric of the common best candidate. WMT14 applies the same 0.90 threshold to the joint BLEU/chrF event. From this near-best set, we choose the candidate with the lowest F, breaking exact ties by higher calibration-quality score and then earlier query order.

Test evaluation. The selected schedule is evaluated on the complete XSum test set of 11,334 examples and the complete WMT14 De→En test set of 3,003 examples. XSum reports ROUGE-1/2/L F1 with stemming. WMT14 reports SacreBLEU corpus BLEU and chrF. The native baseline draws one LogitNormal grid with parameters $(-1.5, 0.8)$ for each ordered batch of 64; the StepTune grid is fixed across all batches. Paired comparisons use the same test examples and initial latents. We use 16,384 item-bootstrap replicates, based on per-example ROUGE for XSum and sentence-level translation scores for WMT14. Repeated-trigram rate, length, EOS rate, and empty-output rate are recorded as generation diagnostics.

Search time for the deployed schedules. The ELF NFE–16 schedules evaluated in Table 1 were obtained with 160 search queries. On a single NVIDIA B200, search takes 0.57 GPU-minutes for WMT14 with 16 calibration examples and 6.79 GPU-minutes for XSum with 64 examples. We measure wall-clock time with the model resident in accelerator memory, including schedule rollouts, metric computation, and Bayesian optimization bookkeeping. The measurements exclude checkpoint loading, reference-cache construction, and held-out evaluation. The selected schedules are frozen and reused, so this one-time search cost is amortized over subsequent generation.

Inference latency and throughput. Latency and throughput are measured on one NVIDIA B200 with the model resident in accelerator memory. Latency uses batch size one; throughput uses batch size 64. Five warm-up blocks precede 30 measured blocks, and method order is randomized across measured blocks. We synchronize the accelerator around every timed block.

The timing boundary includes input transfer, condition encoding, latent construction, sampling, model decoding, and CPU text decoding. It excludes model loading, dataset I/O, input tokenization, grid lookup and transfer, metric computation, and offline calibration. Every compared schedule uses the same timing boundary. Offline search cost is reported separately above and is not included in per-example latency or throughput.

LD3 adaptation. LD3 is evaluated at deployment NFE 16, 32, and 64 with the same teacher NFEs used above. For each task, its 64-example calibration set is split equally between parameter fitting and checkpoint selection. The adaptation optimizes interval boundaries and model-evaluation times while preserving conditioning positions and restricting terminal ℓ_2 error to the generated suffix. The first two epochs optimize only the boundaries; the next five optimize both grids. Boundary parameters use RMSprop with learning rate 0.005 and momentum 0.9. Evaluation-time offsets use SGD with learning rate 0.1. The gradient norm is clipped at 1. Optimization and checkpoint selection use only terminal latent ℓ_2 error. Image-specific LPIPS and relaxed-initial-noise operations are omitted.

Objective-ablation variants. In Section 5.3, A-only uses latent error A to guide search and selects the candidate with the lowest A; KL-only applies the same rule to decoder KL. Pure-F searches and selects using the joint discrepancy F, whereas Pure-Q searches for and selects the highest calibration quality Q. Full uses the complete StepTune procedure: F guides search under quality constraints, and the lowest-F schedule is selected from the feasible near-best-quality set.

B.6 Generation experiments on AURORA-LM-S

These experiments assess StepTune on a second continuous DLM architecture using the released AURORA-LM-S checkpoints. Both conditional and unconditional generation use the native tan-d schedule as the baseline. We retain the model-specific sampling and decoding settings below.

XSum sampling and calibration. We use the released XSum checkpoint at training step 300,000 and the official conditional preprocessing. Candidate schedules use Euler ODE sampling at NFE 16. The paired reference uses the same frozen model and the native tan-d schedule at NFE 32. We retain classifier-free-guidance scale 2.0, the guidance noise window $[0.9, 1.0]$, repainting, and a generated suffix of 64 tokens. Calibration uses 16 examples and 160 queries in the six-dimensional four-region boundary/allocation space. The two components of F are normalized by their calibration means under native tan-d sampling at NFE 16.

The initial design contains the native schedule and 15 scrambled Sobol proposals. Subsequent proposals use a Gaussian-process lower confidence bound with an RBF kernel, length scale 0.22, jitter 10^{-6}, and exploration coefficient 1.8. Proposals are selected from a global Sobol pool augmented with local proposals around strong observed schedules. The search is guided by F, with measured quality violations penalized during calibration.

XSum quality gates and final selection. For this supporting evaluation, we use fixed baseline-relative quality gates: each of ROUGE-1, ROUGE-2, and ROUGE-L must be at least 85% of its value under native tan-d sampling at NFE 16, and the empty-output rate must not exceed 5%. Among the fully evaluated schedules satisfying these gates, we select the one with the lowest F. No additional near-best-quality tolerance is applied in this experiment. We freeze the selected schedule and reuse it in five independent evaluation runs of 1,000 XSum examples, reporting mean ROUGE-1/2/L. The comparison in Table 2b uses the native tan-d schedule at NFE 32.

OpenWebText sampling and calibration. We use the released OpenWebText checkpoint at training step 250,000. Candidate and reference rollouts use the official SDE-DPM++ sampler with self-conditioning-guidance scale 4.0 and `eos_wrap`. Deployment uses NFE 16; the reference uses the native tan-d schedule at NFE 64. Search uses 16 calibration generations, 40 queries, and the same six-dimensional four-region search space. Initialization contains the native schedule and 15 scrambled Sobol points, followed by RBF Gaussian-process lower-confidence-bound proposals. Search generations use a maximum length of 256 tokens, while final evaluation uses the official 1,024-token protocol.

Relative to the native NFE–16 calibration baseline, the perplexity ratio must not exceed 1.35, the unigram-entropy decrease must not exceed 0.15, the repeated-trigram-rate increase must not exceed 0.05, and the empty-output rate must not exceed 0.01. These quality requirements constrain the F-guided search. The selected schedule is frozen before evaluation.

OpenWebText evaluation. We evaluate the frozen schedule and native NFE–16 baseline with five independent seeds and 1,000 generations per seed. Perplexity is computed with GPT-2-Large, and unigram entropy uses the GPT-2 tokenizer. MAUVE compares the generated text with the first 1,000 held-out OpenWebText documents following the released AURORA evaluation recipe. We report the mean across the five runs and additionally record repeated-trigram and empty-output rates as diagnostics.

C The Structure of the Calibrated Schedules

We summarize the eleven deployed schedules for the main generation results using the fraction of evaluation times before $t = 0.10$, 0.25, and 0.50. Here, t increases from the initial-noise endpoint 0

Table 6: **Temporal allocation of the deployed schedules.** Entries are percentages of the N model-evaluation times strictly below each threshold, computed from the saved float32 grids.

Model	Task	NFE	$t < 0.10$	$t < 0.25$	$t < 0.50$
ELF-B	XSum	16	93.8	100.0	100.0
ELF-B	XSum	32	90.6	90.6	90.6
ELF-B	XSum	64	92.2	92.2	92.2
ELF-B	WMT14	16	93.8	93.8	93.8
ELF-B	WMT14	32	78.1	81.3	87.5
ELF-B	WMT14	64	90.6	93.8	96.9
ELF-B	OpenWebText	8	25.0	50.0	87.5
ELF-B	OpenWebText	16	18.8	75.0	100.0
ELF-B	OpenWebText	32	15.6	78.1	96.9
AURORA-LM-S	XSum	16	37.5	87.5	87.5
AURORA-LM-S	OpenWebText	16	6.3	25.0	81.3

to the data endpoint 1. For an NFE-N grid, we count its N left endpoints $t_0, \ldots, t_{N-1}$, including $t_0 = 0$ and excluding the terminal endpoint $t_N = 1$. The final interval still ends at 1, even when all evaluation times lie earlier.

The ELF conditional schedules consistently concentrate evaluations near the beginning of the trajectory: 78.1–93.8% occur before $t = 0.10$ across the six settings. Their precise placements differ, but the preference for early evaluations recurs across tasks and budgets. The unconditional schedules also allocate 50.0–78.1% of evaluations before $t = 0.25$, although these were selected from an already early-skewed LogitNormal candidate family.

AURORA-LM-S shows a more task-dependent pattern. Its XSum schedule places 87.5% of evaluations before $t = 0.25$. On OpenWebText, only 25.0% lie before that threshold, while 56.3% lie in $[0.25, 0.50)$. Thus, early concentration is common in the tested settings, particularly on ELF, but is not a universal shape shared by all selected schedules. These observations describe the calibrated grids; they do not establish a generally optimal allocation rule.

D Additional Related Work

Beyond the methods discussed in the main text, several studies optimize sampling through numerical-error objectives. Optimal Stepsize uses dynamic programming to distill timestep schedules from reference trajectories (Pei et al., 2025). HSO combines a global search for initialization with local refinement, using a midpoint-error proxy and a penalty against excessively close timesteps (Zhu et al., 2026). ART formulates time reparameterization as continuous-time control and learns the sampling clock using an Euler-error surrogate and actor–critic updates (Huang et al., 2026).

Other approaches optimize a broader set of sampler parameters or adapt the grid to individual inputs. BNS distills a family of non-stationary solvers, including their evaluation times, against accurate reference solutions (Shaul et al., 2024). DDSS differentiates through sampling and sample-quality scores to optimize a flexible sampler (Watson et al., 2022). INDIS learns instance-dependent discretizations for image and video generation (Yuan et al., 2026).

These studies offer different objectives and search spaces for allocating sampling computation. Our focus is timestep calibration for continuous DLMs, where a decoder maps terminal representations to token distributions. StepTune uses both latent error and actual decoder KL to guide calibration, incorporates language-quality constraints, and reuses one selected schedule across inputs without changing the model or solver.