ENDPOINT-COMPLETE DIFFUSION SAMPLING VIA LOG-SNR BOUNDARY CLOSURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern fast diffusion samplers are typically designed as finite-interval integrators over a prescribed sequence of noise levels. This view reduces discretization error between neighboring nodes, but leaves a separate source of error at the two truncated ends of the sampling trajectory. We revisit diffusion sampling in log-SNR space, where the complete trajectory extends from the low-SNR limit $\lambda \to -\infty$ to the high-SNR limit $\lambda \to +\infty$. Taking these limits reveals an asymmetric boundary structure: the low-SNR endpoint admits a regular signal-prediction closure, whereas the high-SNR endpoint admits a regular noise-prediction closure. Motivated by this asymmetry, we propose endpoint-complete diffusion sampling, which keeps the interior solver unchanged and replaces only the first and last updates with endpoint-specific boundary plug-ins: DPM-FS and DPM-LS. We further connect the same endpoint asymmetry to generalized velocity prediction and show that, under variance normalization, EDM-style residual preconditioning maps non-degenerate linear targets to the same generalized velocity direction up to sign. Experiments on CIFAR-10, AFHQv2, FFHQ, and ImageNet demonstrate improved few-step sampling quality.

1 INTRODUCTION

Diffusion models have become a central framework for generative modeling, from nonequilibrium diffusion processes (Sohl-Dickstein et al., 2015) and denoising diffusion probabilistic models (Ho et al., 2020; Nichol & Dhariwal, 2021) to score-based generative modeling (Song & Ermon, 2019; 2020; Song et al., 2020) and large-scale image synthesis systems (Dhariwal & Nichol, 2021; Rombach et al., 2022; Saharia et al., 2022; Yang et al., 2023). A central practical challenge is sampling efficiency. Recent work has improved fast diffusion sampling through deterministic samplers (Song et al., 2021), high-order ODE solvers (Karras et al., 2022; Zhang & Chen, 2023; Lu et al., 2022; 2025), predictor-corrector frameworks (Zhao et al., 2023), empirical model statistics (Zheng et al., 2023), and schedule design (Sabour et al., 2024; Chen et al., 2024). A common abstraction behind many of these methods is to first choose a finite set of noise levels and then approximate the probability-flow trajectory between neighboring nodes (Song et al., 2020). This finite-interval view has led to increasingly accurate interior updates, but it leaves open how the two truncated ends of the trajectory should be closed.

In this paper, we revisit diffusion sampling in log-SNR space. The log-SNR coordinate has been used to organize continuous-time objectives, training weights and preconditioning (Kingma et al., 2021; Karras et al., 2022; Hang et al., 2023). In the continuous log-SNR parameterization, the diffusion trajectory connects the low-SNR limit $\lambda \to -\infty$ to the high-SNR limit $\lambda \to +\infty$. Practical samplers truncate both ends, leaving two boundary regions outside the finite interior interval. We formulate their treatment as boundary closures. Taking the endpoint limits reveals an intrinsic asymmetry between the two prediction forms: the signal-form update remains regular at the low-SNR boundary, while the noise-form update remains regular at the high-SNR boundary. The first update therefore requires signal prediction, and the last requires noise prediction.

This asymmetry imposes a requirement on the model supporting the sampler. Ordinary pure signal- or noise-prediction models do not support the endpoint prediction required by this design. Recovering signal from a noise predictor amplifies its error by $e^{-\lambda}$ near the low-SNR end; recovering noise from

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.

a signal predictor amplifies its error by e^{λ} near the high-SNR end. The two closures thus require a predictor whose effective target changes from signal at low SNR to noise at high SNR.

We identify this structure through generalized velocity, the angular component of a rescaled signal–noise mixture. Under admissible rescalings, its endpoint limits are finite and proportional to signal at low SNR and noise at high SNR, providing the prediction structure required by the closures. We further show that, under variance normalization, optimal skip projection and residual normalization map non-degenerate linear targets to the same generalized velocity direction up to sign. This connects the required asymmetric prediction to EDM-style preconditioning (Karras et al., 2022): a model exposed as a signal predictor can internally learn an endpoint-compatible residual. The equivalence concerns residual targets; learned prediction errors still depend on training.

Building on this model structure, we propose endpoint-complete diffusion sampling in log-SNR space. DPM-FS closes the low-SNR boundary from $\lambda \to -\infty$ to a finite transition λ_L in signal form, while DPM-LS closes the high-SNR boundary from λ_R to $\lambda \to +\infty$ in noise form. An estimated first step (EFS) supplies a signal anchor using a dataset or conditional mean. A compatible finite-interval solver transports the state between the transitions without changing its interior update formula.

Conventional sampling schedules are designed for finite-interval interior solvers, with the sampling endpoints typically fixed by the schedule itself and no explicit allocation to endpoint solvers. They are therefore not directly suited to the combination of endpoint and interior solvers. Our experiments show that FSLS can effectively handle wider boundary SNR intervals, allowing the interior integration interval to be reduced while maintaining or improving sample quality. This motivates Extend Polynomial, an empirical schedule that assigns more of the boundary regions to the endpoint solvers. We additionally adapt GITS (Chen et al., 2024) to obtain schedules better matched to the combined sampler. As in the original GITS, the selected schedules are precomputed and stored in a schedule bank, which provides an optimized configuration for each sampling setting.

We evaluate six interior solvers across CIFAR-10, AFHQv2, FFHQ, and ImageNet-512, including EDM2-S, RAE (Zheng et al., 2026), and PixelDiT (Yu et al., 2026). At NFE=5 with FS caching for multistep methods, GITS+FSLS improves over both the Polynomial baseline and ordinary GITS for all six solvers on all three ImageNet-512 backbones. These gains reflect the combined boundary treatment and node allocation. For the SDXL (Podell et al., 2024) COCO-30k benchmark, we use a signal prediction at $t = 999$, counted as one NFE, as a practical substitute for dataset-based EFS. This finite-time initialization retains approximation error relative to the exact endpoint anchor. Despite this approximation, both FSLS configurations improve FID and CLIPScore over Polynomial on COCO-30k at total NFE=10 for all three evaluated solvers.

Our contributions are summarized as follows:

- We identify the asymmetric regular structure of diffusion boundaries in log-SNR space and the corresponding requirement for signal prediction at low SNR and noise prediction at high SNR.
- We establish generalized velocity and normalized residual preconditioning as a prediction foundation for this endpoint design, connecting the required asymmetry to practical model parameterizations.
- We derive DPM-FS and DPM-LS and construct a practical sampler with EFS, an empirical Extend Polynomial schedule, and a schedule bank obtained through adapted GITS, preserving compatible interior update rules.
- We demonstrate few-step improvements across solver and model families, analyze the interaction between endpoint treatment and schedules.

2 Background and Related Work

2.1 Log-SNR formulation

Diffusion models define a family of noisy states by gradually mixing signal with a reference noise variable. Let $\boldsymbol{x} \sim p_{\text{data}}$ be a clean signal sample, and let $\boldsymbol{\epsilon} \sim p_{\text{noise}}$ be an independent noise sample.

At log-SNR coordinate λ, we write

$$\boldsymbol{x}_\lambda = \alpha_\lambda \boldsymbol{x} + \sigma_\lambda \boldsymbol{\epsilon}, \qquad \lambda = \log \frac{\alpha_\lambda}{\sigma_\lambda}. \tag{1}$$

This representation separates the signal direction $\boldsymbol{x}$, the noise direction $\boldsymbol{\epsilon}$, and the scalar log-SNR coordinate used throughout the paper.

When the reference noise is Gaussian, the probability-flow solution between two finite nodes s and t can be written in the log-SNR parameterization used in fast ODE samplers (Lu et al., 2022; 2025) as

$$\boldsymbol{x}_t = \frac{\alpha_t}{\alpha_s}\boldsymbol{x}_s - \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-\lambda} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda, \lambda) \mathrm{d}\lambda, \tag{2}$$

$$= \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + \sigma_t \int_{\lambda_s}^{\lambda_t} e^{\lambda} \boldsymbol{x}_\theta(\boldsymbol{x}_\lambda, \lambda) \mathrm{d}\lambda. \tag{3}$$

Here $\boldsymbol{\epsilon}_\theta$ in Eq. (2) is the noise prediction model, while $\boldsymbol{x}_\theta$ in Eq. (3) is the signal prediction model.

2.2 Prediction parameterizations and preconditioning

Diffusion models can be trained with several related prediction targets. Noise prediction estimates the injected noise $\boldsymbol{\epsilon}$ and is related to score prediction by a known scale factor under Gaussian perturbations (Ho et al., 2020; Song et al., 2020). Signal (or data) prediction estimates the clean sample $\boldsymbol{x}$ (Song et al., 2023; Li & He, 2025). Velocity prediction is commonly written in the VP convention as

$$\bar{\boldsymbol{v}}_\lambda = -\sigma_\lambda \boldsymbol{x} + \alpha_\lambda \boldsymbol{\epsilon}, \tag{4}$$

Progressive distillation, Imagen Video, and Simple Diffusion report empirical benefits from using v-prediction (Salimans & Ho, 2022; Ho et al., 2022; Hoogeboom et al., 2023). Its endpoint behavior is asymmetric: the target becomes signal-directed at low SNR and noise-directed at high SNR. For endpoint prediction, these limiting roles are the essential requirement; generalized velocity representations can preserve them under suitable rescalings.

The model's output parameterization also depends on preconditioning. For example, an EDM-style signal predictor (Karras et al., 2022) combines a skip connection with a scaled network output:

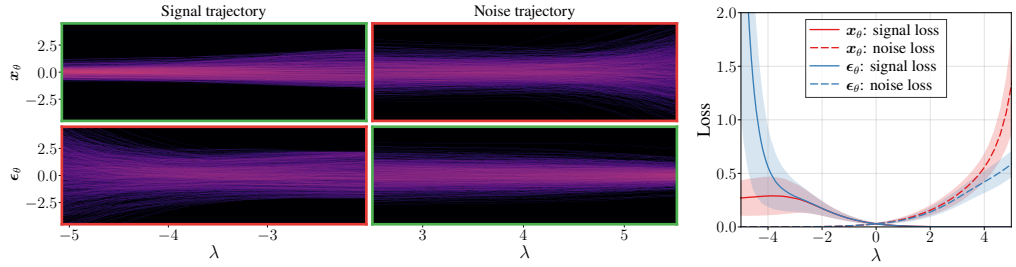
$$\boldsymbol{x}_\theta(\boldsymbol{x}_\lambda, \lambda) = c_{\mathrm{skip}}(\lambda)\boldsymbol{x}_\lambda + c_{\mathrm{out}}(\lambda) F_\theta\big(c_{\mathrm{in}}(\lambda)\boldsymbol{x}_\lambda, \lambda\big).$$

Here c_{in}, c_{skip}, and c_{out} control input scaling, the skip connection, and output scaling, respectively. The network learns a residual relative to the skip prediction, whose target and endpoint behavior can differ from those of the model's final signal output. The quantity learned by the network, the model's output, and the prediction form used by the solver must therefore be distinguished.

2.3 Sampling solvers, schedules, and boundary treatment

Fast ODE samplers transport the state between neighboring finite noise levels. Solver design improves these local updates through higher-order integration, predictor–corrector constructions, or model statistics (Karras et al., 2022; Lu et al., 2022; 2025; Zhao et al., 2023; Zheng et al., 2023). Sampling schedules specify the finite interval and its intermediate nodes. Polynomial provides an empirical node-placement rule, while Align Your Steps and GITS optimize sampling schedules (Sabour et al., 2024; Chen et al., 2024). GITS uses trajectory-based costs and dynamic programming to precompute schedules, which are stored in a schedule bank and reused during sampling. The choice of sampling schedule depends on the dataset, the trained model, and the solver, and no single schedule is uniformly optimal across all settings (Chen, 2023; Sabour et al., 2024).

Boundary and initialization treatments address additional sources of sampling error. Lin et al. (2024) connect nonzero terminal SNR to a training–inference mismatch and restricted brightness range. Their remedies include zero terminal SNR, $\boldsymbol{v}$-prediction, and starting sampling from the last training timestep. The analytical first step (AFS) in GENIE replaces the first learned-score evaluation with the analytical score of the Gaussian initialization to save one NFE (Dockhorn et al., 2022). OMS adds a learned first-step correction for schedule mismatch and low-frequency control (Hu et al., 2024). Related approaches modify global image statistics through offset noise or color alignment (Guttenberg, 2023; Shum et al., 2025).



(a) (b)

Figure 1: Empirical motivation for asymmetric predictor choice at the two endpoints on CIFAR-10. **(a)** Qualitative endpoint behavior. In the initial sampling regime, recovering signal through the noise predictor is inaccurate. In the terminal regime, recovering noise through the signal predictor becomes divergent. **(b)** The same phenomenon viewed numerically through λ-wise mean-squared errors. The four curves evaluate the recovered signal and noise from a signal-loss model $\boldsymbol{x}_\theta$ and a noise-loss model $\boldsymbol{\epsilon}_\theta$ on forward-noised real images at each λ. Statistics are computed over 50,000 CIFAR-10 samples, and shaded regions denote standard deviation.

3 Endpoint-Complete Sampling in Log-SNR Space

3.1 Endpoint Regularity and Boundary Closures

We close the two boundary intervals in the prediction forms suggested by Figure 1: signal recovery is unstable from a noise predictor near the low-SNR boundary, while noise recovery is unstable from a signal predictor near the high-SNR boundary.

The same asymmetry follows from the endpoint structure of the two ODE forms. The left-boundary update becomes regular after expressing it in signal form and expanding with respect to e^{λ}, whereas the dual noise-form normalization is singular there. At the right boundary, the situation is reversed: the noise-form closure is regular after expanding with respect to $e^{-\lambda}$, while the signal-form normalization becomes singular. We make this point explicit in Appendix A.2.

The derivation follows the finite-interval signal and noise prediction parameterizations used in DPM-Solver and DPM-Solver++ (Lu et al., 2022; 2025), but takes one endpoint of the interval to the corresponding log-SNR limit.

Proposition 1 (Asymptotic boundary closures)**.** *Assume that $\boldsymbol{x}_\theta(\boldsymbol{x}_\lambda, \lambda)$, written along the trajectory as a function of e^{λ}, has bounded derivatives up to order k near $e^{\lambda} = 0$. Assume also that $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda, \lambda)$, written along the trajectory as a function of $e^{-\lambda}$, has bounded derivatives up to order k on $[0, e^{-\lambda_R}]$. Then the left and right boundary updates admit the asymptotic representations*

$$\boldsymbol{x}_{\lambda_L} = \sigma_{\lambda_L}\boldsymbol{\epsilon} + \alpha_{\lambda_L}\tilde{\boldsymbol{x}}_\theta^{(k)} + \mathcal{O}\Big(\alpha_{\lambda_L} e^{k\lambda_L}\Big) \tag{5}$$

and

$$\boldsymbol{x} = \frac{\boldsymbol{x}_{\lambda_R} - \sigma_{\lambda_R}\tilde{\boldsymbol{\epsilon}}_\theta^{(k)}}{\alpha_{\lambda_R}} + \mathcal{O}\Big(e^{-(k+1)\lambda_R}\Big), \tag{6}$$

where the endpoint predictors are defined as

$$\tilde{\boldsymbol{x}}_\theta^{(k)} := \sum_{n=0}^{k-1} \frac{e^{n\lambda_L}}{(n+1)!} \left.\frac{\mathrm{d}^n \boldsymbol{x}_\theta(\boldsymbol{x}_\lambda, \lambda)}{\mathrm{d}(e^{\lambda})^n}\right|_{\lambda \to -\infty} \tag{7}$$

and

$$\tilde{\boldsymbol{\epsilon}}_\theta^{(k)} := \sum_{n=0}^{k-1} \frac{(-e^{-\lambda_R})^n}{(n+1)!} \left.\frac{\mathrm{d}^n \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda, \lambda)}{\mathrm{d}(e^{-\lambda})^n}\right|_{\lambda = \lambda_R}. \tag{8}$$

Proposition 1 gives the mathematical form of the asymmetric predictor choice suggested by Figure 1. Eq. (5) provides a signal-form asymptotic closure for the first step, whereas Eq. (6) provides a noise-form asymptotic closure for the last step.

The left-boundary closure also provides a mechanism for the low-frequency and exposure failures observed in noise-form samplers. Near $\lambda \to -\infty$, recovering signal through a noise predictor requires a singular normalization, so small noise-prediction errors can be amplified before the interior solver begins. We visualize this trajectory-level effect and its color-distribution consequence in Appendix C.

3.2 Endpoint-compatible prediction and preconditioning

The two closures require a prediction target that is signal-directed at low SNR and noise-directed at high SNR. To identify this structure, consider the mixture path $\boldsymbol{x}_\lambda = \alpha_\lambda \boldsymbol{x} + \sigma_\lambda \boldsymbol{\epsilon}$ for a fixed signal–noise pair $(\boldsymbol{x}, \boldsymbol{\epsilon})$ (Salimans & Ho, 2022). For a positive, continuously differentiable rescaling r_λ, define $\boldsymbol{z}_\lambda := \boldsymbol{x}_\lambda / r_\lambda$ and the generalized velocity target $\boldsymbol{v}_\lambda := (-\sigma_\lambda \boldsymbol{x} + \alpha_\lambda \boldsymbol{\epsilon})/r_\lambda$. The angle of the mixing-coefficient pair $(\alpha_\lambda, \sigma_\lambda)$ is

$$\phi_\lambda := \arctan\left(\frac{\sigma_\lambda}{\alpha_\lambda}\right) = \arctan(e^{-\lambda}). \tag{9}$$

This coefficient-space geometry describes the mixture path and its prediction target.

Proposition 2 (Generalized rescaled angular dynamics)**.** *For positive, continuously differentiable α_λ, σ_λ, and r_λ, define*

$$s_\lambda := \sqrt{\alpha_\lambda^2 + \sigma_\lambda^2}, \qquad \kappa_\lambda := \frac{\mathrm{d}\log(s_\lambda / r_\lambda)}{\mathrm{d}\phi_\lambda}. \tag{10}$$

For the fixed signal–noise pair, the rescaled mixture path $\boldsymbol{z}_\lambda$ satisfies

$$\frac{\mathrm{d}\boldsymbol{z}_\lambda}{\mathrm{d}\phi_\lambda} = \kappa_\lambda \boldsymbol{z}_\lambda + \boldsymbol{v}_\lambda. \tag{11}$$

Proposition 2 separates angular change in the mixing coefficients from radial scaling: $\boldsymbol{v}_\lambda$ supplies the angular component, while $\kappa_\lambda \boldsymbol{z}_\lambda$ accounts for the remaining scale change. Appendix A.4 visualizes this geometry.

Different choices of r_λ produce different targets; matching the endpoint scales preserves the required asymmetric limits.

Corollary 1 (Endpoint limits under endpoint-scale matching)**.** *Suppose that r_λ satisfies*

$$\frac{r_\lambda}{\sigma_\lambda} \to c_- > 0 \quad \textit{as} \quad \lambda \to -\infty, \qquad \frac{r_\lambda}{\alpha_\lambda} \to c_+ > 0 \quad \textit{as} \quad \lambda \to +\infty. \tag{12}$$

Then the generalized target $\boldsymbol{v}_\lambda$ admits finite endpoint limits given by

$$\boldsymbol{v}_\lambda \to -\frac{1}{c_-}\boldsymbol{x} \quad \textit{as} \quad \lambda \to -\infty, \qquad \boldsymbol{v}_\lambda \to \frac{1}{c_+}\boldsymbol{\epsilon} \quad \textit{as} \quad \lambda \to +\infty. \tag{13}$$

Thus $\boldsymbol{v}_\lambda$ is asymptotically signal-directed at low SNR and noise-directed at high SNR.

Let $\boldsymbol{v}_\theta(\boldsymbol{x}_\lambda, \lambda)$ predict this generalized target. Inverting the mixture representation gives the two predictions needed by the closures:

$$\boldsymbol{x}_\theta = \frac{\alpha_\lambda \boldsymbol{x}_\lambda - \sigma_\lambda r_\lambda \boldsymbol{v}_\theta}{s_\lambda^2}, \qquad \boldsymbol{\epsilon}_\theta = \frac{\sigma_\lambda \boldsymbol{x}_\lambda + \alpha_\lambda r_\lambda \boldsymbol{v}_\theta}{s_\lambda^2}. \tag{14}$$

Under Corollary 1, the coefficient of $\boldsymbol{v}_\theta$ in signal recovery tends to $-c_-$ at low SNR, and its coefficient in noise recovery tends to c_+ at high SNR. These finite coefficients connect generalized velocity prediction to the asymmetric closures in Section 3.1. For the circular choice $r_\lambda = s_\lambda$, the radial term vanishes and generalized $\boldsymbol{v}_\lambda$ reduces to the standard VP velocity under $\alpha_\lambda^2 + \sigma_\lambda^2 = 1$.

Residual preconditioning. Let $\boldsymbol{t}_\lambda = p_\lambda \boldsymbol{x} + q_\lambda \boldsymbol{\epsilon}$, with independent, zero-mean signal and noise of variances $\sigma_{\mathrm{data}}^2 \mathbf{I}$ and $\sigma_{\mathrm{noise}}^2 \mathbf{I}$, where $\sigma_{\mathrm{data}}, \sigma_{\mathrm{noise}} > 0$. EDM-style preconditioning (Karras et al., 2022) scales the input by $c_{\mathrm{in}} = (\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2)^{-1/2}$ and trains the normalized residual

$$F_{\mathrm{target}} = (\boldsymbol{t}_\lambda - c_{\mathrm{skip}} \boldsymbol{x}_\lambda)/c_{\mathrm{out}}, \tag{15}$$

where c_{skip} is the optimal linear skip coefficient and c_{out} is the residual standard deviation.

Proposition 3 (Residual preconditioning induces a canonical residual direction). *Let $\boldsymbol{t}_\lambda = p_\lambda \boldsymbol{x} + q_\lambda \boldsymbol{\epsilon}$ and assume $p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda \neq 0$ at finite λ. Then the optimal skip coefficient is*

$$c_{\text{skip}}(\lambda) = \frac{p_\lambda \alpha_\lambda \sigma_{\text{data}}^2 + q_\lambda \sigma_\lambda \sigma_{\text{noise}}^2}{\alpha_\lambda^2 \sigma_{\text{data}}^2 + \sigma_\lambda^2 \sigma_{\text{noise}}^2}, \tag{16}$$

and the normalized residual target can be written as

$$F_{\text{target}} = \operatorname{sgn}(p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda) \frac{\sigma_\lambda \sigma_{\text{noise}}^2 \boldsymbol{x} - \alpha_\lambda \sigma_{\text{data}}^2 \boldsymbol{\epsilon}}{\sigma_{\text{data}} \sigma_{\text{noise}} \sqrt{\alpha_\lambda^2 \sigma_{\text{data}}^2 + \sigma_\lambda^2 \sigma_{\text{noise}}^2}}. \tag{17}$$

Therefore, after optimal skip projection and output normalization, every non-degenerate linear target induces the same λ-dependent residual direction up to sign. At an endpoint where the residual scale vanishes, the normalized target is interpreted through its limit.

For $\sigma_{\text{data}} = \sigma_{\text{noise}} = 1$ and $r_\lambda = s_\lambda$, this gives

$$F_{\text{target}} = -\operatorname{sgn}(p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda) \boldsymbol{v}_\lambda. \tag{18}$$

Thus the normalized residual labels share the angular direction identified above. When the sign is constant, as for the standard signal, noise, and VP velocity targets, the labels differ only by an overall sign. This equivalence concerns residual labels; training weights, optimization, and solver parameterization can still affect the learned dynamics and sampling error. Proofs and scale-conversion details are given in Appendix A.

3.3 Practical endpoint-complete sampling

Let $\lambda_1 < \cdots < \lambda_N$ be the finite nodes, with transition nodes $\lambda_2 = \lambda_L$ and $\lambda_{N-1} = \lambda_R$. DPM-FS constructs $\boldsymbol{x}_{\lambda_2}$ in signal form, a standard interior solver advances it to $\boldsymbol{x}_{\lambda_{N-1}}$, and DPM-LS completes sampling in noise form.

Estimated first step. Diffusion training can be viewed as jointly learning denoising tasks at different noise levels (Hang et al., 2023). As SNR decreases, the noisy input provides less sample-specific signal information. Under squared-error training, the Bayes-optimal signal prediction is the conditional mean, which approaches $\mathbb{E}[\boldsymbol{x} \mid \mathcal{C}]$ as the input loses its signal information, where $\mathcal{C}$ denotes the condition. This motivates the expectation that sufficiently optimized signal predictions become less dependent on the noise realization near the low-SNR endpoint. The signal-directed endpoint limit of generalized velocity is compatible with this conditional-mean prediction. Our estimated first step (EFS), denoted $\hat{\boldsymbol{x}}^{\text{EFS}}$, uses a precomputed dataset or class mean; a learned conditional estimator extends the construction to text prompts (Appendix B.1).

Second-order first step. DPM-FS-2 takes the discrete form

$$\boldsymbol{x}_{\lambda_2} = \sigma_{\lambda_2} \boldsymbol{\epsilon} + \alpha_{\lambda_2} \left[\left(1 - \frac{e^{\lambda_2 - \lambda_1}}{2}\right) \hat{\boldsymbol{x}}^{\text{EFS}} + \frac{e^{\lambda_2 - \lambda_1}}{2} \boldsymbol{x}_\theta(\boldsymbol{x}_{\lambda_1}, \lambda_1) \right], \tag{19}$$

where $\boldsymbol{x}_{\lambda_1} = \alpha_{\lambda_1} \hat{\boldsymbol{x}}^{\text{EFS}} + \sigma_{\lambda_1} \boldsymbol{\epsilon}$ is the auxiliary state constructed from EFS at $\lambda_1 < \lambda_2$. The bracketed combination gives the discrete second-order endpoint predictor $\tilde{\boldsymbol{x}}_\theta^{(2)}$ corresponding to Eq. (5).

Second-order last step. DPM-LS-2 takes the discrete form

$$\begin{aligned} \boldsymbol{x} = \frac{\boldsymbol{x}_{\lambda_{N-1}}}{\alpha_{\lambda_{N-1}}} - \frac{\sigma_{\lambda_{N-1}}}{\alpha_{\lambda_{N-1}}} \Bigg[& \frac{\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_N}, \lambda_N)}{2(1 - e^{\lambda_{N-1} - \lambda_N})} \\ & + \left(1 - \frac{1}{2(1 - e^{\lambda_{N-1} - \lambda_N})}\right) \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_{N-1}}, \lambda_{N-1}) \Bigg], \end{aligned} \tag{20}$$

where the auxiliary state at $\lambda_N > \lambda_{N-1}$ is

$$\boldsymbol{x}_{\lambda_N} = \frac{\alpha_{\lambda_N}}{\alpha_{\lambda_{N-1}}} \boldsymbol{x}_{\lambda_{N-1}} + \left(\sigma_{\lambda_N} - \frac{\alpha_{\lambda_N}}{\alpha_{\lambda_{N-1}}} \sigma_{\lambda_{N-1}}\right) \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_{N-1}}, \lambda_{N-1}).$$

The bracketed combination gives the discrete second-order endpoint predictor $\tilde{\boldsymbol{\epsilon}}_\theta^{(2)}$ corresponding to Eq. (6). Together, these modules form FSLS. Appendix A.3 derives these finite-difference approximations, and Appendix B.6 specifies the implementation.

First-step history. For multistep solvers, optional FS caching associates the combined endpoint prediction $\tilde{\boldsymbol{x}}_\theta^{(2)}$ from Eq. (19) with the first auxiliary node λ_1 to initialize the history. The stored quantity and state follow the solver's signal- or noise-form convention. This option changes history initialization while preserving the interior update formula.

3.4 Sampling schedules for endpoint–interior coupling

Conventional schedules allocate finite nodes to an interior solver and usually determine its endpoints as part of that allocation. With FSLS, the transition nodes also determine how much of the trajectory is handled by each boundary closure. Our transition experiments show that FSLS can handle wider boundary regions while maintaining effective sampling, allowing a shorter interior integration interval. The schedule must therefore be chosen for the combined endpoint and interior solver under the total evaluation budget.

Extend Polynomial. We use an empirical allocation in the polynomial coordinate $u = e^{-\lambda/\rho}$, with reference bounds $u_{\max} > u_{\min} > 0$. For a reference node count $A \geq 5$, set

$$u_L = u_{\max} + \frac{\varkappa_L}{A-1}(u_{\min} - u_{\max}), \tag{21}$$

$$u_R = u_{\min} + \frac{\varkappa_R}{A-1}(u_{\max} - u_{\min}). \tag{22}$$

The parameters $\varkappa_L$ and $\varkappa_R$ move the transitions inward from the reference bounds, assigning more of the boundary regions to FSLS. Interior nodes are uniform between u_L and u_R; the auxiliary nodes are $u_L + h$ and $u_R - h$, where $h = (u_L - u_R)/(A-3)$, subject to positive, ordered nodes. The reference node count, actual finite-node count, and NFE have distinct roles; their correspondence and the model-specific settings are given in Appendix D.

GITS. We adapt GITS (Chen et al., 2024) to select transitions, interior-node placement, and auxiliary gaps for the complete endpoint-enabled sampler. A constrained offline optimization matches terminal student and teacher image features, evaluating each candidate with its endpoint modules and multistep history. The selected schedules are stored in a bank for each model, solver, NFE, and cache setting, then reused unchanged during sampling. Appendix D.3 specifies the feature-matching objective, candidate selection, and calibration costs.

4 Experiments

4.1 Experimental setup

We evaluate DDIM, Heun2nd, DPM-Solver++2S/2M, UniPC, and iPNDM-V (Song et al., 2021; Karras et al., 2022; Lu et al., 2025; Zhao et al., 2023; Zhang & Chen, 2023). We train the CIFAR-10 (Krizhevsky et al., 2009) (32^2), AFHQv2 (Choi et al., 2020; Karras et al., 2021) and FFHQ (Karras et al., 2019) (64^2) models using the EDM architecture and training recipe. On these datasets and ImageNet-512 (Deng et al., 2009) with EDM2-S (Karras et al., 2024), we evaluate NFE $\in \{5, 7, 9, 11, 13, 15\}$. We additionally evaluate RAE (Zheng et al., 2026) and PixelDiT (Yu et al., 2026) on ImageNet-512 at NFE=5. Text-to-image evaluation uses SDXL (Podell et al., 2024) on COCO-30k (Lin et al., 2014) at 1024^2 and total NFE=10.

For the image benchmarks, FID-50k (Heusel et al., 2017) is computed from 50,000 generated images, using `torch-fidelity` (Obukhov et al., 2020) for the three low-resolution datasets and the official EDM2 Inception/reference-statistics pipeline for ImageNet-512. SDXL uses 30,000 image–caption pairs and reports both FID and CLIPScore (Hessel et al., 2021). NFE counts evaluations of the sampling vector field, including the endpoint modules. FSLS uses DPM-FS-2 and DPM-LS-2 throughout. Checkpoints, guidance, evaluation protocols, and offline costs are detailed in Appendix E.1 and Appendix D.3.

4.2 Few-step sampling across budgets and model families

Across the budget sweeps in Tables 3–6, FSLS supports few-step sampling with both first-order and higher-order interior solvers when coupled to a suitable schedule. The gains obtained without

Table 1: ImageNet-512 FID-50k at NFE=5. FS cache is enabled for multistep FSLS configurations. Baseline uses Polynomial; Baseline and ordinary GITS omit FSLS. Bold and underlining mark the best and second-best FID within each row.

Model	Solver	Baseline	Extend + FSLS	GITS	GITS + FSLS
EDM2-S (Karras et al., 2024)	DDIM	83.99	8.69	54.83	**7.94**
	Heun2nd	208.43	**7.41**	251.25	8.54
	DPM++2S	92.96	7.09	104.78	**6.44**
	DPM++2M	49.18	6.43	33.63	**5.65**
	UniPC	48.31	**6.54**	30.11	7.31
	iPNDM-V	60.65	6.48	13.03	**5.87**
RAE (Zheng et al., 2026)	DDIM	30.06	**9.88**	49.36	10.02
	Heun2nd	239.09	16.65	236.86	**10.43**
	DPM++2S	29.51	**10.12**	26.02	11.30
	DPM++2M	24.70	**8.82**	53.17	10.03
	UniPC	24.59	**9.81**	52.24	11.05
	iPNDM-V	29.02	**9.98**	51.87	10.99
PixelDiT (Yu et al., 2026)	DDIM	33.63	14.02	19.58	**9.39**
	Heun2nd	200.72	64.87	218.86	**16.60**
	DPM++2S	35.95	13.43	36.45	**8.92**
	DPM++2M	28.70	11.83	20.73	**7.95**
	UniPC	28.76	12.21	20.87	**7.36**
	iPNDM-V	31.87	14.44	19.88	**8.77**

changing their interior update rules support treating boundary closure as a complementary part of sampler design.

Table 1 extends this evaluation to three ImageNet-512 backbones. The effectiveness of the same endpoint construction across these models supports its use with different interior solvers, while the varying schedule preferences indicate that node allocation must account for both the model and solver.

4.3 Predictor asymmetry and schedule coupling

Matching the prediction form to the endpoint. The predictor-switching experiment in Figure 2 supports the roles assigned to the two closures: signal prediction at low SNR and noise prediction at high SNR. Training-formulation comparisons are given in Appendix E.5.

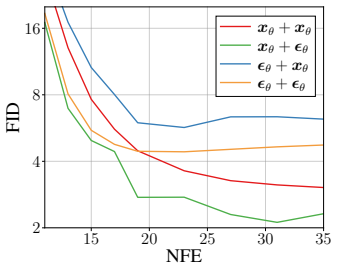
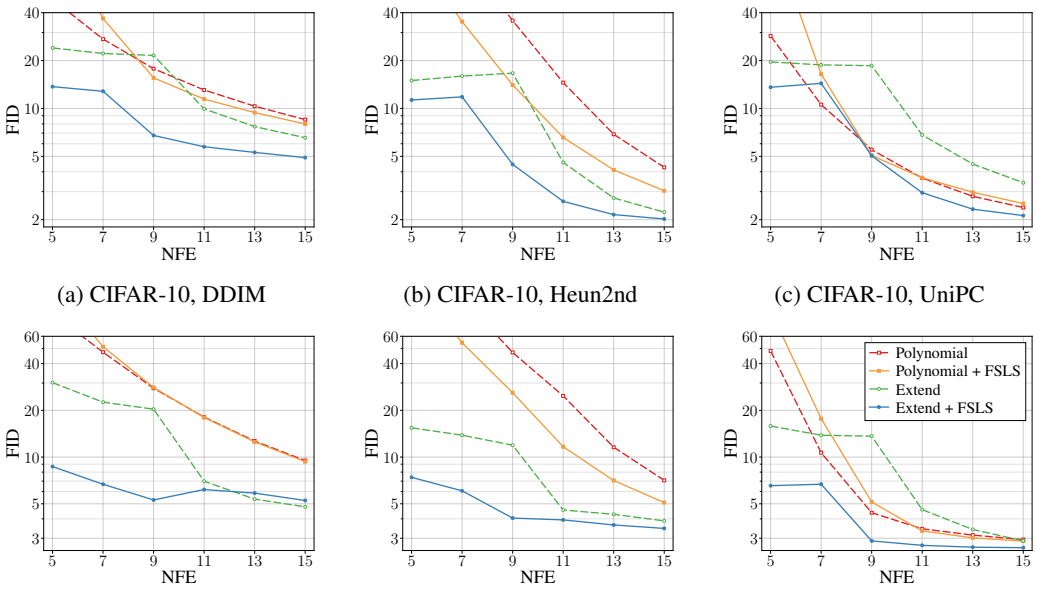


Figure 2: Predictor combinations on CIFAR-10 with Heun2nd, switching models at $\lambda = 0$.

Allocating a fixed budget. Figure 3 shows why endpoint modules and schedules need to be evaluated together. Under a fixed budget, boundary calls replace some interior evaluations. The degradation observed when inserting FSLS into Polynomial is consistent with a loss of interior resolution under the inherited node allocation. Extend assigns larger boundary regions to FSLS and shortens the interior integration interval. The combined improvement supports allocating the trajectory according to the distinct roles of the endpoint and interior solvers.

The different responses to Extend alone, including Heun2nd's favorable response, show that node allocation also depends on the interior update. This motivates adapting GITS to evaluate the complete sampler, including its endpoint modules and history, so that the schedule bank reflects their joint behavior. Further schedule and endpoint ablations are reported in Appendix D and Appendix E.



(a) CIFAR-10, DDIM (b) CIFAR-10, Heun2nd (c) CIFAR-10, UniPC

(d) ImageNet-512, DDIM (e) ImageNet-512, Heun2nd (f) ImageNet-512, UniPC

Figure 3: FID-50k versus NFE for Polynomial and Extend, with and without FSLS. ImageNet-512 uses EDM2-S. FS cache is enabled for UniPC endpoint variants.

4.4 Text-to-image sampling

We evaluate DPM++2M, UniPC, and iPNDM-V with SDXL base 1.0 at 1024×1024 and CFG 5. In this COCO-30k evaluation, a clean-signal prediction at native $t = 999$ provides a finite-time approximation to EFS, retaining initialization error. FSLS uses this one-NFE initialization plus nine subsequent sampler calls, with FS cache enabled; endpoint-free baselines also use ten calls. The full protocol is in Appendix E.1.

Table 2: SDXL on COCO-30k at total NFE=10. Baseline is Polynomial without FSLS. FSLS uses the $t = 999$ initialization and FS cache. Bold and underlining mark the best and second-best values of each metric within each solver.

Solver	Baseline		Extend + FSLS		GITS		GITS + FSLS	
	FID ↓	CLIP ↑	FID ↓	CLIP ↑	FID ↓	CLIP ↑	FID ↓	CLIP ↑
DPM++2M	14.24	26.29	<u>14.05</u>	**26.86**	15.90	26.11	**12.88**	<u>26.53</u>
UniPC	14.24	26.25	<u>13.96</u>	**26.95**	15.67	26.02	**12.84**	<u>26.49</u>
iPNDM-V	14.44	26.27	<u>13.87</u>	**26.85**	14.94	26.06	**12.69**	<u>26.53</u>

Table 2 shows that boundary treatment remains useful with a finite-time signal anchor whose evaluation consumes part of the sampling budget. This extends the construction to text-to-image sampling when a dataset-based EFS is unavailable.

5 Conclusion

We formulated diffusion sampling as finite interior integration with asymmetric endpoint closures, using signal prediction at low SNR and noise prediction at high SNR. Generalized velocity supports this asymmetry, while Extend Polynomial and adapted GITS allocate the sampling budget between endpoint and interior solvers. Experiments across solvers and model families support the effectiveness of this joint design. We encourage developers of large diffusion models to prepare EFS during training. This supplies the initial signal estimate without an extra denoising-network evaluation, leaving the limited NFE budget for subsequent sampling updates.

AI USE STATEMENT

We used generative AI tools to polish the manuscript and revise its structure, assist in writing selected non-core components of the supplementary code, and organize code and draft explanatory text in Jupyter notebooks. We also used these tools to cross-check our mathematical reasoning. The authors take full responsibility for the final content of this work, including the mathematical claims, code, and reported results.

REFERENCES

Defang Chen, Zhenyu Zhou, Can Wang, Chunhua Shen, and Siwei Lyu. On the trajectory regularity of ode-based diffusion sampling. *arXiv preprint arXiv:2405.11326*, 2024.

Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8188–8197, 2020.

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

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

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Genie: Higher-order denoising diffusion solvers. *Advances in Neural Information Processing Systems*, 35:30150–30166, 2022.

Nicholas Guttenberg. Diffusion with offset noise, 2023. URL `https://www.crosslabs.org/blog/diffusion-with-offset-noise`.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7441–7451, 2023.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7514–7528, 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pp. 13213–13232. PMLR, 2023.

Minghui Hu, Jianbin Zheng, Chuanxia Zheng, Chaoyue Wang, Dacheng Tao, and Tat-Jen Cham. One more step: A versatile plug-and-play module for rectifying diffusion schedule flaws and enhancing low-frequency controls. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7331–7340, 2024.

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

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in neural information processing systems*, 34:852–863, 2021.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24174–24184. IEEE, 2024.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

LAION. Releasing re-laion-5b: transparent iteration on laion-5b with additional safety fixes. `https://laion.ai/blog/relaion-5b/`, 2024. Accessed: 30 aug, 2024.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 5404–5411, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

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

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

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Anton Obukhov, Maximilian Seitzer, Po-Wei Wu, Semen Zhydenko, Jonathan Kyl, and Elvis Yu-Jing Lin. High-fidelity performance metrics for generative models in pytorch, 2020. URL `https://github.com/toshas/torch-fidelity`. Version: 0.4.0, DOI: 10.5281/zenodo.3786539.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, volume 2024, pp. 1862–1874, 2024.

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

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. *arXiv preprint arXiv:2404.14507*, 2024.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Ka Chun Shum, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Color alignment in diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 28446–28455, 2025.

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

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

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

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39, 2023.

Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14273–14282, 2026.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:49842–49869, 2023.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, volume 2026, pp. 35791–35820, 2026.

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

A Additional derivations

A.1 Proof of Proposition 1

Proof. Starting from the left-boundary update corresponding to Eq. (3), we let the previous node tend to $-\infty$, i.e., $\lambda_{i-1} \to -\infty$ and set $\lambda_i = \lambda_L$. Then,

$$\boldsymbol{x}_{\lambda_L} = \lim_{\lambda_{i-1}\to-\infty} \Big[\frac{\sigma_{\lambda_L}}{\sigma_{\lambda_{i-1}}}\boldsymbol{x}_{\lambda_{i-1}} + \sigma_{\lambda_L}\int_{-\infty}^{\lambda_L} e^{\lambda}\boldsymbol{x}_\theta(\boldsymbol{x}_\lambda,\lambda)\mathrm{d}\lambda\Big] \tag{23}$$

$$= \sigma_{\lambda_L}\Big[\frac{\boldsymbol{x}_{-\infty}}{\sigma_{-\infty}} + \int_{-\infty}^{\lambda_L} e^{\lambda}\boldsymbol{x}_\theta(\boldsymbol{x}_\lambda,\lambda)\mathrm{d}\lambda\Big]. \tag{24}$$

Since

$$\frac{\boldsymbol{x}_{-\infty}}{\sigma_{-\infty}} = \lim_{\lambda\to-\infty}\frac{\alpha_\lambda\boldsymbol{x}+\sigma_\lambda\boldsymbol{\epsilon}}{\sigma_\lambda} = \lim_{\lambda\to-\infty} e^{\lambda}\boldsymbol{x}+\boldsymbol{\epsilon} = \boldsymbol{\epsilon}, \tag{25}$$

we obtain

$$\boldsymbol{x}_{\lambda_L} = \sigma_{\lambda_L}\Big[\boldsymbol{\epsilon} + \int_{-\infty}^{\lambda_L} e^{\lambda}\boldsymbol{x}_\theta(\boldsymbol{x}_\lambda,\lambda)\mathrm{d}\lambda\Big]. \tag{26}$$

This integral does not admit a closed-form expression. We therefore introduce the change of variable

$$u := e^{\lambda}, \qquad h_{\boldsymbol{x}}(u) := \boldsymbol{x}_\theta(\boldsymbol{x}_{\log u}, \log u). \tag{27}$$

Since $e^{\lambda}\mathrm{d}\lambda = \mathrm{d}u$, we obtain

$$\boldsymbol{x}_{\lambda_L} = \sigma_{\lambda_L}\Big[\boldsymbol{\epsilon} + \int_0^{e^{\lambda_L}} h_{\boldsymbol{x}}(u)\mathrm{d}u\Big]. \tag{28}$$

Applying a Taylor expansion of the integrand at $u = 0$, then

$$\boldsymbol{x}_{\lambda_L} = \sigma_{\lambda_L}\Big[\boldsymbol{\epsilon} + \int_0^{e^{\lambda_L}} \Big(\sum_{n=0}^{k-1}\frac{h_{\boldsymbol{x}}^{(n)}(0)}{n!}u^n + \mathcal{O}(u^k)\Big)\mathrm{d}u\Big] \tag{29}$$

$$= \sigma_{\lambda_L}\Big[\boldsymbol{\epsilon} + \sum_{n=0}^{k-1}\frac{h_{\boldsymbol{x}}^{(n)}(0)}{(n+1)!}e^{(n+1)\lambda_L}\Big] + \mathcal{O}\Big(\sigma_{\lambda_L}e^{(k+1)\lambda_L}\Big) \tag{30}$$

$$= \sigma_{\lambda_L}\Big[\boldsymbol{\epsilon} + \sum_{n=0}^{k-1}\frac{e^{(n+1)\lambda_L}}{(n+1)!}\left.\frac{\mathrm{d}^n\boldsymbol{x}_\theta(\boldsymbol{x}_\lambda,\lambda)}{\mathrm{d}(e^{\lambda})^n}\right|_{\lambda\to-\infty}\Big] + \mathcal{O}(\alpha_{\lambda_L}e^{k\lambda_L}). \tag{31}$$

Using $e^{\lambda_L} = \alpha_{\lambda_L}/\sigma_{\lambda_L}$, we can rewrite Eq. (31) as

$$\boldsymbol{x}_{\lambda_L} = \sigma_{\lambda_L}\boldsymbol{\epsilon} + \alpha_{\lambda_L}\tilde{\boldsymbol{x}}_\theta^{(k)} + \mathcal{O}(\alpha_{\lambda_L}e^{k\lambda_L}), \tag{32}$$

where

$$\tilde{\boldsymbol{x}}_\theta^{(k)} := \sum_{n=0}^{k-1}\frac{e^{n\lambda_L}}{(n+1)!}\left.\frac{\mathrm{d}^n\boldsymbol{x}_\theta(\boldsymbol{x}_\lambda,\lambda)}{\mathrm{d}(e^{\lambda})^n}\right|_{\lambda\to-\infty} \tag{33}$$

denotes the k-th order endpoint predictor of the signal.

Starting from the right-boundary update corresponding to Eq. (2), we let $\lambda_i \to +\infty$ and set $\lambda_{i-1} = \lambda_R$. Then

$$\boldsymbol{x}_{+\infty} = \lim_{\lambda_i\to+\infty}[\frac{\alpha_{\lambda_i}}{\alpha_{\lambda_R}}\boldsymbol{x}_{\lambda_R} - \alpha_{\lambda_i}\int_{\lambda_R}^{+\infty} e^{-\lambda}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda,\lambda)\mathrm{d}\lambda] \tag{34}$$

$$= \alpha_{+\infty}[\frac{\boldsymbol{x}_{\lambda_R}}{\alpha_{\lambda_R}} - \int_{\lambda_R}^{+\infty} e^{-\lambda}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda,\lambda)\mathrm{d}\lambda]. \tag{35}$$

Noting that

$$\frac{\boldsymbol{x}_{+\infty}}{\alpha_{+\infty}} = \lim_{\lambda\to+\infty}\frac{\alpha_\lambda\boldsymbol{x}+\sigma_\lambda\epsilon}{\alpha_\lambda} = \boldsymbol{x} + \lim_{\lambda\to+\infty} e^{-\lambda}\boldsymbol{\epsilon} = \boldsymbol{x}, \tag{36}$$

from which we can obtain

$$\boldsymbol{x} = \frac{\boldsymbol{x}_{\lambda_R}}{\alpha_{\lambda_R}} - \int_{\lambda_R}^{+\infty} e^{-\lambda} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda, \lambda) \mathrm{d}\lambda. \tag{37}$$

Now we introduce the change of variable

$$u := e^{-\lambda}, \qquad h_{\boldsymbol{\epsilon}}(u) := \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{-\log u}, -\log u). \tag{38}$$

Since $e^{-\lambda}\mathrm{d}\lambda = -\mathrm{d}u$, we obtain

$$\boldsymbol{x} = \frac{\boldsymbol{x}_{\lambda_R}}{\alpha_{\lambda_R}} - \int_0^{e^{-\lambda_R}} h_{\boldsymbol{\epsilon}}(u) \mathrm{d}u. \tag{39}$$

Applying a Taylor expansion at $u = e^{-\lambda_R}$ yields

$$\boldsymbol{x} = \frac{\boldsymbol{x}_{\lambda_R}}{\alpha_{\lambda_R}} - \int_0^{e^{-\lambda_R}} \Big[\sum_{n=0}^{k-1} \frac{h_{\boldsymbol{\epsilon}}^{(n)}(e^{-\lambda_R})}{n!} (u - e^{-\lambda_R})^n + \mathcal{O}\Big((u - e^{-\lambda_R})^k\Big) \Big] \mathrm{d}u \tag{40}$$

$$= \frac{\boldsymbol{x}_{\lambda_R}}{\alpha_{\lambda_R}} + \sum_{n=0}^{k-1} \frac{h_{\boldsymbol{\epsilon}}^{(n)}(e^{-\lambda_R})}{(n+1)!} (-e^{-\lambda_R})^{n+1} + \mathcal{O}\Big(e^{-(k+1)\lambda_R}\Big) \tag{41}$$

$$= \frac{\boldsymbol{x}_{\lambda_R}}{\alpha_{\lambda_R}} + \sum_{n=0}^{k-1} \frac{(-e^{-\lambda_R})^{n+1}}{(n+1)!} \left. \frac{\mathrm{d}^n \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda, \lambda)}{\mathrm{d}(e^{-\lambda})^n} \right|_{\lambda=\lambda_R} + \mathcal{O}\Big(e^{-(k+1)\lambda_R}\Big). \tag{42}$$

We can also write it as

$$\boldsymbol{x} = \frac{\boldsymbol{x}_{\lambda_R} - \sigma_{\lambda_R} \tilde{\boldsymbol{\epsilon}}_\theta^{(k)}}{\alpha_{\lambda_R}} + \mathcal{O}\Big(e^{-(k+1)\lambda_R}\Big), \tag{43}$$

where

$$\tilde{\boldsymbol{\epsilon}}_\theta^{(k)} := \sum_{n=0}^{k-1} \frac{(-e^{-\lambda_R})^n}{(n+1)!} \left. \frac{\mathrm{d}^n \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda, \lambda)}{\mathrm{d}(e^{-\lambda})^n} \right|_{\lambda=\lambda_R} \tag{44}$$

denotes the k-th order endpoint predictor of the noise.

This completes the proof. □

A.2 Endpoint regularity of signal-form and noise-form closures

The proof in Appendix A.1 already establishes the left- and right-boundary closures in Eq. (32) and Eq. (43). Here we only explain why these two closures are the natural ones, and why the opposite predictor choices are singular at the corresponding endpoints.

For the left boundary, Appendix A.1 starts from the signal-form update in Eq. (24) and rewrites it as:

$$\sigma_{\lambda_L} \Big[\boldsymbol{\epsilon} + \int_{-\infty}^{\lambda_L} e^{\lambda} \boldsymbol{x}_\theta(\boldsymbol{x}_\lambda, \lambda) \mathrm{d}\lambda \Big]. \tag{45}$$

This representation is regular for two reasons. First, by $\boldsymbol{x}_{-\infty}/\sigma_{-\infty} = \boldsymbol{\epsilon}$, the boundary term is finite. Second, after the change of variable $u = e^{\lambda}$, integral becomes

$$\int_0^{e^{\lambda_L}} h_{\boldsymbol{x}}(u) \mathrm{d}u, \tag{46}$$

which is regular at $u = 0$ whenever $h_{\boldsymbol{x}}$ is smooth there. This is exactly why the left-boundary closure in Appendix A.1 is naturally expressed in signal form and expanded with respect to e^{λ}.

By contrast, the opposite normalization would be based on $\boldsymbol{x}_\lambda/\alpha_\lambda = \boldsymbol{x} + e^{-\lambda}\boldsymbol{\epsilon}$. As $\lambda \to -\infty$, the second term is generally unbounded, so the corresponding boundary quantity is singular. At the same endpoint, the kernel $e^{-\lambda}$ is also singular rather than integrable. Therefore, a noise-form closure is not natural at the initial boundary.

The right boundary is completely dual: the noise-form representation is regular because $\boldsymbol{x}_\lambda/\alpha_\lambda$ admits a finite endpoint limit and the change of variable $u = e^{-\lambda}$ transforms the integral into one over

$(0, e^{-\lambda_R}]$, while the signal-form normalization $\boldsymbol{x}_\lambda/\sigma_\lambda = e^\lambda \boldsymbol{x} + \boldsymbol{\epsilon}$ becomes singular as $\lambda \to +\infty$. Hence the terminal boundary is naturally written in noise form rather than signal form.

Therefore, the asymmetric predictor choice is already encoded in the endpoint structure of the formulas proved in Appendix A.1: the left boundary is regular in signal form but singular in noise form, whereas the right boundary is regular in noise form but singular in signal form. Proposition 1 formalizes this asymmetry through the endpoint predictors $\tilde{\boldsymbol{x}}_\theta^{(k)}$ and $\tilde{\boldsymbol{\epsilon}}_\theta^{(k)}$.

A.3 Practical first- and second-order endpoint predictors

First-order endpoint predictors. For the first-order left boundary closure, the predictor reduces to

$$\tilde{\boldsymbol{x}}_\theta^{(1)} = \boldsymbol{x}_\theta(\boldsymbol{x}_{-\infty}, -\infty). \tag{47}$$

For the first-order right boundary closure, the predictor reduces to

$$\tilde{\boldsymbol{\epsilon}}_\theta^{(1)} = \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_N}, \lambda_N). \tag{48}$$

Second-order endpoint predictors. For the second-order left boundary closure, the target node is λ_2, and λ_1 is used as the auxiliary node. From Eq. (33), we have

$$\tilde{\boldsymbol{x}}_\theta^{(2)} = \boldsymbol{x}_\theta(\boldsymbol{x}_{-\infty}, -\infty) + \frac{e^{\lambda_2}}{2} \left.\frac{\mathrm{d}\boldsymbol{x}_\theta(\boldsymbol{x}_\lambda, \lambda)}{\mathrm{d}(e^\lambda)}\right|_{\lambda \to -\infty} \tag{49}$$

$$\approx \boldsymbol{x}_\theta(\boldsymbol{x}_{-\infty}, -\infty) + \frac{e^{\lambda_2}}{2} \frac{\boldsymbol{x}_\theta(\boldsymbol{x}_{\lambda_1}, \lambda_1) - \boldsymbol{x}_\theta(\boldsymbol{x}_{-\infty}, -\infty)}{e^{\lambda_1}} \tag{50}$$

$$= \frac{e^{\lambda_2 - \lambda_1}}{2} \boldsymbol{x}_\theta(\boldsymbol{x}_{\lambda_1}, \lambda_1) + \Big(1 - \frac{e^{\lambda_2 - \lambda_1}}{2}\Big) \boldsymbol{x}_\theta(\boldsymbol{x}_{-\infty}, -\infty). \tag{51}$$

For the second-order right boundary closure, the target node is λ_{N-1}, and λ_N is used as the auxiliary node. From Eq. (44), we have

$$\tilde{\boldsymbol{\epsilon}}_\theta^{(2)} = \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_{N-1}}, \lambda_{N-1}) - \frac{e^{-\lambda_{N-1}}}{2} \left.\frac{\mathrm{d}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\lambda, \lambda)}{\mathrm{d}(e^{-\lambda})}\right|_{\lambda \to \lambda_{N-1}} \tag{52}$$

$$\approx \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_{N-1}}, \lambda_{N-1}) - \frac{e^{-\lambda_{N-1}}}{2} \frac{\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_{N-1}}, \lambda_{N-1}) - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_N}, \lambda_N)}{e^{-\lambda_{N-1}} - e^{-\lambda_N}} \tag{53}$$

$$= \frac{1}{2(1 - e^{\lambda_{N-1} - \lambda_N})} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_N}, \lambda_N) + \Big(1 - \frac{1}{2(1 - e^{\lambda_{N-1} - \lambda_N})}\Big) \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_{N-1}}, \lambda_{N-1}). \tag{54}$$

A.4 Proof of Proposition 2

Proof. Let

$$\boldsymbol{x}_\lambda = \alpha_\lambda \boldsymbol{x} + \sigma_\lambda \boldsymbol{\epsilon}, \qquad \boldsymbol{z}_\lambda := \frac{\boldsymbol{x}_\lambda}{r_\lambda}, \qquad \boldsymbol{v}_\lambda := \frac{-\sigma_\lambda \boldsymbol{x} + \alpha_\lambda \boldsymbol{\epsilon}}{r_\lambda}, \tag{55}$$

where $(\boldsymbol{x}, \boldsymbol{\epsilon})$ is a fixed signal–noise pair and α_λ, σ_λ, and r_λ are positive and continuously differentiable; define

$$s_\lambda := \sqrt{\alpha_\lambda^2 + \sigma_\lambda^2}. \tag{56}$$

We first write the transformation from $(\boldsymbol{x}, \boldsymbol{\epsilon})$ to $(\boldsymbol{z}_\lambda, \boldsymbol{v}_\lambda)$ in matrix form:

$$\begin{pmatrix} \boldsymbol{z}_\lambda \\ \boldsymbol{v}_\lambda \end{pmatrix} = \frac{1}{r_\lambda} \begin{pmatrix} \alpha_\lambda & \sigma_\lambda \\ -\sigma_\lambda & \alpha_\lambda \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{\epsilon} \end{pmatrix}. \tag{57}$$

Its inverse is

$$\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{\epsilon} \end{pmatrix} = \frac{r_\lambda}{s_\lambda^2} \begin{pmatrix} \alpha_\lambda & -\sigma_\lambda \\ \sigma_\lambda & \alpha_\lambda \end{pmatrix} \begin{pmatrix} \boldsymbol{z}_\lambda \\ \boldsymbol{v}_\lambda \end{pmatrix}, \tag{58}$$

that is

$$\boldsymbol{x} = \frac{r_\lambda}{s_\lambda^2}(\alpha_\lambda \boldsymbol{z}_\lambda - \sigma_\lambda \boldsymbol{v}_\lambda), \qquad \boldsymbol{\epsilon} = \frac{r_\lambda}{s_\lambda^2}(\sigma_\lambda \boldsymbol{z}_\lambda + \alpha_\lambda \boldsymbol{v}_\lambda). \tag{59}$$

Next, differentiate

$$\boldsymbol{z}_\lambda = \frac{\alpha_\lambda}{r_\lambda}\boldsymbol{x} + \frac{\sigma_\lambda}{r_\lambda}\boldsymbol{\epsilon}. \tag{60}$$

Since $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$ are independent of λ, we obtain

$$\mathrm{d}\boldsymbol{z}_\lambda = \Big(\frac{\mathrm{d}\alpha_\lambda}{r_\lambda} - \frac{\alpha_\lambda \mathrm{d}r_\lambda}{r_\lambda^2}\Big)\boldsymbol{x} + \Big(\frac{\mathrm{d}\sigma_\lambda}{r_\lambda} - \frac{\sigma_\lambda \mathrm{d}r_\lambda}{r_\lambda^2}\Big)\boldsymbol{\epsilon}. \tag{61}$$

Substituting Eq. (59) above yields

$$\mathrm{d}\boldsymbol{z}_\lambda = \frac{1}{s_\lambda^2}\Big(\mathrm{d}\alpha_\lambda - \alpha_\lambda \frac{\mathrm{d}r_\lambda}{r_\lambda}\Big)(\alpha_\lambda \boldsymbol{z}_\lambda - \sigma_\lambda \boldsymbol{v}_\lambda) + \frac{1}{s_\lambda^2}\Big(\mathrm{d}\sigma_\lambda - \sigma_\lambda \frac{\mathrm{d}r_\lambda}{r_\lambda}\Big)(\sigma_\lambda \boldsymbol{z}_\lambda + \alpha_\lambda \boldsymbol{v}_\lambda) \tag{62}$$

$$= \Big(\frac{\alpha_\lambda \mathrm{d}\alpha_\lambda + \sigma_\lambda \mathrm{d}\sigma_\lambda}{s_\lambda^2} - \frac{\mathrm{d}r_\lambda}{r_\lambda}\Big)\boldsymbol{z}_\lambda + \frac{\alpha_\lambda \mathrm{d}\sigma_\lambda - \sigma_\lambda \mathrm{d}\alpha_\lambda}{s_\lambda^2}\boldsymbol{v}_\lambda. \tag{63}$$

Because

$$\mathrm{d}\log s_\lambda = \frac{\mathrm{d}s_\lambda^2}{2s_\lambda^2} = \frac{\mathrm{d}(\alpha_\lambda^2 + \sigma_\lambda^2)}{2s_\lambda^2} = \frac{\alpha_\lambda \mathrm{d}\alpha_\lambda + \sigma_\lambda \mathrm{d}\sigma_\lambda}{s_\lambda^2}, \tag{64}$$

Eq. (63) simplifies to

$$\mathrm{d}\boldsymbol{z}_\lambda = \Big(\mathrm{d}\log s_\lambda - \frac{\mathrm{d}r_\lambda}{r_\lambda}\Big)\boldsymbol{z}_\lambda + \frac{\alpha_\lambda \mathrm{d}\sigma_\lambda - \sigma_\lambda \mathrm{d}\alpha_\lambda}{s_\lambda^2}\boldsymbol{v}_\lambda \tag{65}$$

$$= \mathrm{d}\log\Big(\frac{s_\lambda}{r_\lambda}\Big)\boldsymbol{z}_\lambda + \frac{\alpha_\lambda \mathrm{d}\sigma_\lambda - \sigma_\lambda \mathrm{d}\alpha_\lambda}{s_\lambda^2}\boldsymbol{v}_\lambda. \tag{66}$$

Now define

$$\phi_\lambda := \arctan(e^{-\lambda}). \tag{67}$$

Differentiating ϕ_λ, we obtain

$$\mathrm{d}\phi_\lambda = \frac{\alpha_\lambda \mathrm{d}\sigma_\lambda - \sigma_\lambda \mathrm{d}\alpha_\lambda}{\alpha_\lambda^2 + \sigma_\lambda^2} = \frac{\alpha_\lambda \mathrm{d}\sigma_\lambda - \sigma_\lambda \mathrm{d}\alpha_\lambda}{s_\lambda^2}. \tag{68}$$

Therefore,

$$\mathrm{d}\boldsymbol{z}_\lambda = \mathrm{d}\log\Big(\frac{s_\lambda}{r_\lambda}\Big)\boldsymbol{z}_\lambda + \boldsymbol{v}_\lambda \mathrm{d}\phi_\lambda. \tag{69}$$

Dividing both sides by $\mathrm{d}\phi_\lambda$ gives

$$\frac{\mathrm{d}\boldsymbol{z}_\lambda}{\mathrm{d}\phi_\lambda} = \frac{\mathrm{d}\log(s_\lambda/r_\lambda)}{\mathrm{d}\phi_\lambda}\boldsymbol{z}_\lambda + \boldsymbol{v}_\lambda = \kappa_\lambda \boldsymbol{z}_\lambda + \boldsymbol{v}_\lambda, \tag{70}$$

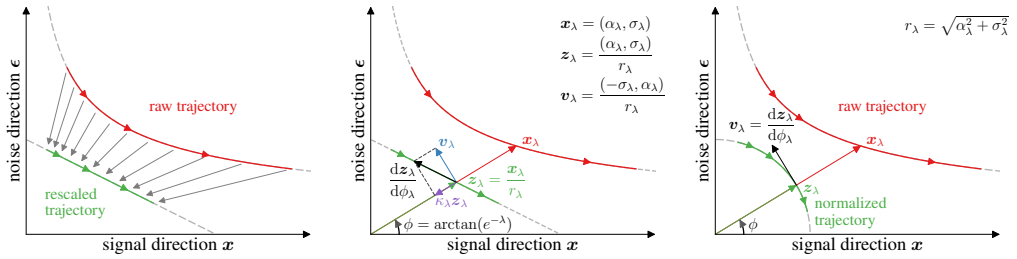
where

$$\kappa_\lambda := \frac{\mathrm{d}\log(s_\lambda/r_\lambda)}{\mathrm{d}\phi_\lambda}. \tag{71}$$

This completes the proof. □

For completeness, we may also write

$$\frac{\mathrm{d}\phi_\lambda}{\mathrm{d}\lambda} = \frac{-e^{-\lambda}}{1 + e^{-2\lambda}} = -\frac{1}{e^{\lambda} + e^{-\lambda}} = -\frac{\alpha_\lambda \sigma_\lambda}{\alpha_\lambda^2 + \sigma_\lambda^2}. \tag{72}$$



(a) (b) (c)

Figure 4: Coefficient-space geometry of the generalized $\boldsymbol{v}$ target for a fixed signal–noise pair. **(a)** In the unnormalized coefficient path, angular change and radial scaling are entangled. **(b)** A general rescaling separates the dynamics into a directional term and a residual radial term. **(c)** For $r_\lambda = s_\lambda = \sqrt{\alpha_\lambda^2 + \sigma_\lambda^2}$, the radial term vanishes, and generalized $\boldsymbol{v}$ is the derivative of the rescaled mixture path with respect to the coefficient angle.

A.5 Proof of Corollary 1

Proof. Under the assumptions

$$\frac{r_\lambda}{\sigma_\lambda} \to c_- > 0 \quad \text{as} \quad \lambda \to -\infty, \qquad \frac{r_\lambda}{\alpha_\lambda} \to c_+ > 0 \quad \text{as} \quad \lambda \to +\infty, \tag{73}$$

the endpoint limits stated in the main text follow directly.

As $\lambda \to -\infty$, we have $\alpha_\lambda/\sigma_\lambda = e^\lambda \to 0$. Therefore,

$$\boldsymbol{v}_\lambda = \frac{-\sigma_\lambda \boldsymbol{x} + \alpha_\lambda \boldsymbol{\epsilon}}{r_\lambda} = \frac{-\boldsymbol{x} + (\alpha_\lambda/\sigma_\lambda)\boldsymbol{\epsilon}}{r_\lambda/\sigma_\lambda} \to -\frac{1}{c_-}\boldsymbol{x}. \tag{74}$$

As $\lambda \to +\infty$, we have $\sigma_\lambda/\alpha_\lambda = e^{-\lambda} \to 0$. Therefore,

$$\boldsymbol{v}_\lambda = \frac{-\sigma_\lambda \boldsymbol{x} + \alpha_\lambda \boldsymbol{\epsilon}}{r_\lambda} = \frac{-(\sigma_\lambda/\alpha_\lambda)\boldsymbol{x} + \boldsymbol{\epsilon}}{r_\lambda/\alpha_\lambda} \to \frac{1}{c_+}\boldsymbol{\epsilon}. \tag{75}$$

This completes the proof. □

A.6 Input variance normalization

Since $\boldsymbol{x}_\lambda = \alpha_\lambda \boldsymbol{x} + \sigma_\lambda \boldsymbol{\epsilon}$, $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$ are independent, we have

$$\mathrm{Var}(\boldsymbol{x}_\lambda) = \alpha_\lambda^2 \mathrm{Var}(\boldsymbol{x}) + \sigma_\lambda^2 \mathrm{Var}(\boldsymbol{\epsilon}) = \alpha_\lambda^2 \sigma_{\text{data}}^2 + \sigma_\lambda^2 \sigma_{\text{noise}}^2. \tag{76}$$

Choosing $c_{\text{in}}(\lambda)$ such that $\mathrm{Var}(c_{\text{in}}(\lambda)\boldsymbol{x}_\lambda) = 1$ yields

$$c_{\text{in}}(\lambda) = \frac{1}{\sqrt{\alpha_\lambda^2 \sigma_{\text{data}}^2 + \sigma_\lambda^2 \sigma_{\text{noise}}^2}}. \tag{77}$$

A.7 Proof of Proposition 3

Proof. For finite λ, assume $p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda \neq 0$ and $\sigma_{\text{data}}, \sigma_{\text{noise}} > 0$, so that the residual standard deviation below is positive. Endpoint values of the normalized target are understood as limits. We define the residual loss

$$L(c) = \mathbb{E}[(\boldsymbol{t}_\lambda - c\boldsymbol{x}_\lambda)^2]. \tag{78}$$

Differentiating with respect to c gives

$$\frac{\mathrm{d}L}{\mathrm{d}c} = -2\mathbb{E}[\boldsymbol{x}_\lambda(\boldsymbol{t}_\lambda - c\boldsymbol{x}_\lambda)]. \tag{79}$$

Setting the derivative to zero yields the least-squares optimum

$$c_{\text{skip}}(\lambda) = \frac{\mathbb{E}[\boldsymbol{x}_\lambda \boldsymbol{t}_\lambda]}{\mathbb{E}[\boldsymbol{x}_\lambda^2]} = \frac{\mathrm{Cov}(\boldsymbol{t}_\lambda, \boldsymbol{x}_\lambda)}{\mathrm{Var}(\boldsymbol{x}_\lambda)}. \tag{80}$$

Using $\boldsymbol{t}_\lambda = p_\lambda \boldsymbol{x} + q_\lambda \boldsymbol{\epsilon}$ and $\boldsymbol{x}_\lambda = \alpha_\lambda \boldsymbol{x} + \sigma_\lambda \boldsymbol{\epsilon}$ together with $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{\epsilon}) = \mathbf{0}$, we obtain

$$\mathrm{Cov}(\boldsymbol{t}_\lambda, \boldsymbol{x}_\lambda) = p_\lambda \alpha_\lambda \sigma_{\mathrm{data}}^2 + q_\lambda \sigma_\lambda \sigma_{\mathrm{noise}}^2. \tag{81}$$

Therefore,

$$c_{\mathrm{skip}}(\lambda) = \frac{p_\lambda \alpha_\lambda \sigma_{\mathrm{data}}^2 + q_\lambda \sigma_\lambda \sigma_{\mathrm{noise}}^2}{\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2}. \tag{82}$$

Now define the residual

$$R_\lambda := \boldsymbol{t}_\lambda - c_{\mathrm{skip}} \boldsymbol{x}_\lambda = \Big(p_\lambda - c_{\mathrm{skip}}(\lambda)\alpha_\lambda\Big)\boldsymbol{x} + \Big(q_\lambda - c_{\mathrm{skip}}(\lambda)\sigma_\lambda\Big)\boldsymbol{\epsilon}. \tag{83}$$

A direct simplification yields

$$p_\lambda - c_{\mathrm{skip}}(\lambda)\alpha_\lambda = \sigma_\lambda \sigma_{\mathrm{noise}}^2 \frac{p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda}{\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2} \tag{84}$$

and

$$q_\lambda - c_{\mathrm{skip}}(\lambda)\sigma_\lambda = -\alpha_\lambda \sigma_{\mathrm{data}}^2 \frac{p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda}{\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2}. \tag{85}$$

Hence

$$R_\lambda = \frac{p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda}{\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2} (\sigma_\lambda \sigma_{\mathrm{noise}}^2 \boldsymbol{x} - \alpha_\lambda \sigma_{\mathrm{data}}^2 \boldsymbol{\epsilon}). \tag{86}$$

Introduce the shared direction $\boldsymbol{d}_\lambda := \sigma_\lambda \sigma_{\mathrm{noise}}^2 \boldsymbol{x} - \alpha_\lambda \sigma_{\mathrm{data}}^2 \boldsymbol{\epsilon}$. Then

$$R_\lambda = \frac{p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda}{\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2} \boldsymbol{d}_\lambda. \tag{87}$$

Since $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$ are independent,

$$\mathrm{Var}(\boldsymbol{d}_\lambda) = \sigma_\lambda^2 \sigma_{\mathrm{noise}}^4 \mathrm{Var}(\boldsymbol{x}) + \alpha_\lambda^2 \sigma_{\mathrm{data}}^4 \mathrm{Var}(\boldsymbol{\epsilon}) = \sigma_{\mathrm{data}}^2 \sigma_{\mathrm{noise}}^2 (\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2). \tag{88}$$

Therefore,

$$\mathrm{Var}(R_\lambda) = \frac{(p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda)^2}{(\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2)^2} \mathrm{Var}(\boldsymbol{d}_\lambda) = \frac{(p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda)^2 \sigma_{\mathrm{data}}^2 \sigma_{\mathrm{noise}}^2}{\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2}. \tag{89}$$

Thus the natural output normalization is

$$c_{\mathrm{out}}(\lambda) = \sqrt{\mathrm{Var}(R_\lambda)} = |p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda| \frac{\sigma_{\mathrm{data}} \sigma_{\mathrm{noise}}}{\sqrt{\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2}}. \tag{90}$$

Substituting back, we obtain

$$F_{\mathrm{target}} = \frac{R_\lambda}{c_{\mathrm{out}}(\lambda)} = \mathrm{sgn}(p_\lambda \sigma_\lambda - q_\lambda \alpha_\lambda) \frac{\sigma_\lambda \sigma_{\mathrm{noise}}^2 \boldsymbol{x} - \alpha_\lambda \sigma_{\mathrm{data}}^2 \boldsymbol{\epsilon}}{\sigma_{\mathrm{data}} \sigma_{\mathrm{noise}} \sqrt{\alpha_\lambda^2 \sigma_{\mathrm{data}}^2 + \sigma_\lambda^2 \sigma_{\mathrm{noise}}^2}}. \tag{91}$$

This completes the proof. □

A.8 Precondition in non-residual form

We now relate the residual result above to the usual non-residual $\boldsymbol{v}$ target. Without the residual skip connection, a variance-normalized linear target takes the form

$$F_{\mathrm{target}}(\boldsymbol{t}_\lambda; \lambda) = \frac{\boldsymbol{t}_\lambda}{c_{\mathrm{out}}(\lambda)}, \qquad c_{\mathrm{out}}(\lambda)^2 = \mathrm{Var}(\boldsymbol{t}_\lambda). \tag{92}$$

For the generic linear target $\boldsymbol{t}_\lambda$ with independent $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$, we have $\mathrm{Var}(\boldsymbol{t}_\lambda) = p_\lambda^2 \sigma_{\mathrm{data}}^2 + q_\lambda^2 \sigma_{\mathrm{noise}}^2$, and therefore

$$F_{\mathrm{target}}(\boldsymbol{t}_\lambda; \lambda) = \frac{p_\lambda \boldsymbol{x} + q_\lambda \boldsymbol{\epsilon}}{\sqrt{p_\lambda^2 \sigma_{\mathrm{data}}^2 + q_\lambda^2 \sigma_{\mathrm{noise}}^2}}. \tag{93}$$

In particular, when both data and noise are variance-normalized, $\sigma_{\mathrm{data}} = \sigma_{\mathrm{noise}} = 1$, it reduces to

$$F_{\mathrm{target}}(\boldsymbol{t}_\lambda; \lambda) = \frac{p_\lambda \boldsymbol{x} + q_\lambda \boldsymbol{\epsilon}}{\sqrt{p_\lambda^2 + q_\lambda^2}}. \tag{94}$$

Under the circular rescaling discussed in Section 3.2, choosing $r_\lambda = \sqrt{\alpha_\lambda^2 + \sigma_\lambda^2}$ and $(p_\lambda, q_\lambda) = (-\sigma_\lambda/r_\lambda, \alpha_\lambda/r_\lambda)$, gives

$$F_{\text{target}}(\boldsymbol{t}_\lambda; \lambda) = \frac{-\sigma_\lambda \boldsymbol{x} + \alpha_\lambda \boldsymbol{\epsilon}}{r_\lambda} = \boldsymbol{v}_\lambda. \tag{95}$$

Therefore, in the variance-normalized setting, the residual-normalized target derived in Proposition 3 can be viewed as a special case within this broader non-residual family. Hence EDM-style residual prediction with arbitrary non-degenerate linear targets is directionally equivalent, up to sign, to non-residual prediction of the circular-rescaled velocity target. Moreover, the VP case simply gives the cleaner form $\alpha_\lambda^2 + \sigma_\lambda^2 = 1$, so that $r_\lambda = 1$ and $\boldsymbol{v}_\lambda = -\sigma_\lambda \boldsymbol{x} + \alpha_\lambda \boldsymbol{\epsilon}$.

A.9 Translating raw log-SNR parameters into normalized coordinates

Many λ-dependent hyperparameters reported in prior work are specified in a raw coordinate system defined on the original data scale. Since our experiments are conducted in a variance-normalized framework, these raw-scale quantities must be translated before they can be reused. This appendix derives the resulting shift in the λ-coordinate.

Let prior work parameterize the noisy state as

$$\boldsymbol{x}^{\text{raw}}_{\lambda^{\text{raw}}} = \alpha^{\text{raw}}_{\lambda^{\text{raw}}} \boldsymbol{x}^{\text{raw}} + \sigma^{\text{raw}}_{\lambda^{\text{raw}}} \boldsymbol{\epsilon}^{\text{raw}}, \tag{96}$$

where $\boldsymbol{x}^{\text{raw}}$ denotes data on the original scale and $\boldsymbol{\epsilon}^{\text{raw}}$ denotes the reference noise variable used in that convention. We rewrite them in normalized unit-variance variables as

$$\boldsymbol{x}^{\text{raw}} = \sigma_{\text{data}} \boldsymbol{x}, \qquad \boldsymbol{\epsilon}^{\text{raw}} = \sigma_{\text{noise}} \boldsymbol{\epsilon}. \tag{97}$$

Substituting Eq. (97) into Eq. (96) gives

$$\boldsymbol{x}^{\text{raw}}_{\lambda^{\text{raw}}} = \sigma_{\text{data}} \alpha^{\text{raw}}_{\lambda^{\text{raw}}} \boldsymbol{x} + \sigma_{\text{noise}} \sigma^{\text{raw}}_{\lambda^{\text{raw}}} \boldsymbol{\epsilon}. \tag{98}$$

Now consider any common rescaling of the intermediate state, $\tilde{\boldsymbol{x}}_\lambda = s(\lambda) \boldsymbol{x}^{\text{raw}}_{\lambda^{\text{raw}}}$, which may in general be λ-dependent. This includes input preconditioning factors such as $c_{\text{in}}(\lambda)$. Then

$$\tilde{\boldsymbol{x}}_\lambda = s(\lambda) \sigma_{\text{data}} \alpha^{\text{raw}}_{\lambda^{\text{raw}}} \boldsymbol{x} + s(\lambda) \sigma_{\text{noise}} \sigma^{\text{raw}}_{\lambda^{\text{raw}}} \boldsymbol{\epsilon}. \tag{99}$$

Hence, in the normalized parameterization

$$\tilde{\boldsymbol{x}}_\lambda = \alpha_\lambda \boldsymbol{x} + \sigma_\lambda \boldsymbol{\epsilon}, \tag{100}$$

the coefficients satisfy

$$\alpha_\lambda = s(\lambda) \sigma_{\text{data}} \alpha^{\text{raw}}_{\lambda^{\text{raw}}}, \qquad \sigma_\lambda = s(\lambda) \sigma_{\text{noise}} \sigma^{\text{raw}}_{\lambda^{\text{raw}}}. \tag{101}$$

Therefore,

$$\lambda = \log \frac{\alpha_\lambda}{\sigma_\lambda} = \log \frac{s(\lambda) \sigma_{\text{data}} \alpha^{\text{raw}}_{\lambda^{\text{raw}}}}{s(\lambda) \sigma_{\text{noise}} \sigma^{\text{raw}}_{\lambda^{\text{raw}}}} = \log \frac{\sigma_{\text{data}}}{\sigma_{\text{noise}}} + \log \frac{\alpha^{\text{raw}}_{\lambda^{\text{raw}}}}{\sigma^{\text{raw}}_{\lambda^{\text{raw}}}}. \tag{102}$$

Thus the λ-coordinates are related by

$$\lambda = \log \frac{\sigma_{\text{data}}}{\sigma_{\text{noise}}} + \lambda^{\text{raw}}. \tag{103}$$

This relation serves purely as a coordinate-conversion rule for reusing raw-scale λ-dependent settings from prior work in our variance-normalized framework. In particular, it applies to schedule endpoints and to the parameters of raw-scale λ-sampling distributions. Under the standard convention $\boldsymbol{\epsilon}^{\text{raw}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, equivalently $\sigma_{\text{noise}} = 1$, the conversion reduces to

$$\lambda = \log \sigma_{\text{data}} + \lambda^{\text{raw}}. \tag{104}$$

B Endpoint estimators and implementation

B.1 Estimated first step

At the pure-noise endpoint, the initial noise $\boldsymbol{\epsilon}$ and signal $\boldsymbol{x}$ are independent given the conditioning information $\mathcal{C}$. Under the squared-error objective, the optimal endpoint signal predictor is therefore

$$f^*(\boldsymbol{\epsilon}, \mathcal{C}) = \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{\epsilon}, \mathcal{C}] = \mathbb{E}[\boldsymbol{x} \mid \mathcal{C}]. \tag{105}$$

To see this directly, let $\boldsymbol{\mu}(\mathcal{C}) = \mathbb{E}[\boldsymbol{x} \mid \mathcal{C}]$. The squared-error risk decomposes as

$$\mathbb{E}\|\boldsymbol{x} - f(\boldsymbol{\epsilon}, \mathcal{C})\|_2^2 = \mathbb{E}\|\boldsymbol{x} - \boldsymbol{\mu}(\mathcal{C})\|_2^2 + \mathbb{E}\|f(\boldsymbol{\epsilon}, \mathcal{C}) - \boldsymbol{\mu}(\mathcal{C})\|_2^2. \tag{106}$$

The first term is independent of f, and the second is minimized by the conditional mean. EFS estimates this target from the training data before sampling.

For a discrete condition c, the initial signal estimate is the empirical mean

$$\hat{\boldsymbol{x}}^{\text{EFS}}(c) = \frac{1}{|\mathcal{D}_c|} \sum_{\boldsymbol{x} \in \mathcal{D}_c} \boldsymbol{x}, \tag{107}$$

where $\mathcal{D}_c$ contains the training samples with condition c, expressed in the same signal representation and normalized coordinates as the sampler. The unconditional case uses the full training set. For text conditioning, a learned conditional estimator approximates the mean, as described in Appendix B.2.

In DPM-FS-2, $\hat{\boldsymbol{x}}^{\text{EFS}}$ constructs the auxiliary state $\boldsymbol{x}_{\lambda_1}$, and the finite-node signal prediction refines it into the discrete endpoint predictor $\tilde{\boldsymbol{x}}_\theta^{(2)}$ in Eq. (19). Preparing EFS in advance avoids a denoising-network evaluation for the initial signal estimate, leaving the limited NFE budget for subsequent prediction refinement and integration updates.

B.2 Text-conditioned EFS for SDXL

We implement the conditional-mean construction in Appendix B.1 with a small text-conditioned network for SDXL. It supplies $\hat{\boldsymbol{x}}^{\text{EFS}}$ from the prompt before any denoising-network evaluation.

Prediction target and architecture. SDXL's frozen dual text encoders provide token features $T(\mathcal{C})$ and a pooled feature $p(\mathcal{C})$. The regression target $\boldsymbol{z} \in \mathbb{R}^{4\times128\times128}$ is the frozen VAE posterior mean of a 1024×1024 image, scaled by 0.13025. The estimator predicts a residual around the training-set spatial latent mean $\bar{\boldsymbol{z}}_{\text{train}}$:

$$\hat{\boldsymbol{z}}^{\text{EFS}}(\mathcal{C}) = \bar{\boldsymbol{z}}_{\text{train}} + g_\eta\big(T(\mathcal{C}), p(\mathcal{C})\big), \tag{108}$$

$$\mathcal{L}_{\text{EFS}}(\eta) = \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}^{\text{EFS}}(\mathcal{C}) - \boldsymbol{z}\|_2^2}{4 \cdot 128^2}\right]. \tag{109}$$

The network has approximately 2.89 million trainable parameters and takes only positive-prompt features as input. Two cross-attention blocks map the text features to a learned 16×16 query grid, followed by three upsampling stages that produce the full latent prediction. The output convolution is initialized to zero, so training starts from the spatial mean.

Data and training. We curate Re-LAION high-resolution image–text pairs (LAION, 2024), with equal representation of dark backgrounds, dark scenes, bright backgrounds, and bright scenes.[1] The training, validation, and test sets contain 4,000, 400, and 400 pairs, respectively; images are center-cropped and resized to 1024×1024, and the original English captions are retained. We train EFS for 30 epochs with AdamW, batch size 16, and learning rate 10^{-4}; validation MSE selects the checkpoint at epoch 6. The text encoders and VAE remain frozen, and all mean and normalization statistics use only the training set.

Sampling interface and budget. The output is converted to the sampler's normalized signal coordinates as

$$\hat{\boldsymbol{x}}^{\text{EFS}}(\mathcal{C}) = \frac{\hat{\boldsymbol{z}}^{\text{EFS}}(\mathcal{C}) - \boldsymbol{\mu}_{\text{ch}}}{\sigma_{\text{data}}}, \tag{110}$$

where $\boldsymbol{\mu}_{\text{ch}}$ is the training-set channel mean and σ_{data} includes both posterior-mean variance and expected VAE posterior variance. One EFS call supplies the initial signal estimate; classifier-free guidance is applied in the subsequent UNet evaluations. The small estimator's inference cost is recorded separately from UNet NFE.

[1] Source: Re-LAION high-resolution metadata.

B.3 SDXL SAMPLING EXAMPLES

Figure 5 compares Polynomial, Extend+FSLS, and GITS+FSLS with iPNDM-V at 1024×1024, CFG 5, and ten UNet evaluations. FSLS enables FS caching and calls the learned EFS once, with no native $t = 999$ initialization call. Schedules are frozen before generating these examples. The two scene prompts are taken from Figure 3(d–e) of Lin et al. (2024); the model and sampling settings are those specified here.

All methods use the same prompt suffix and negative prompt. Mean luma is measured on the original encoded RGB pixels in $[0, 255]$ as the image average of $0.2126R + 0.7152G + 0.0722B$. The FSLS configurations produce darker sky samples and whiter eagle backgrounds. These examples illustrate the complete sampler's brightness behavior.

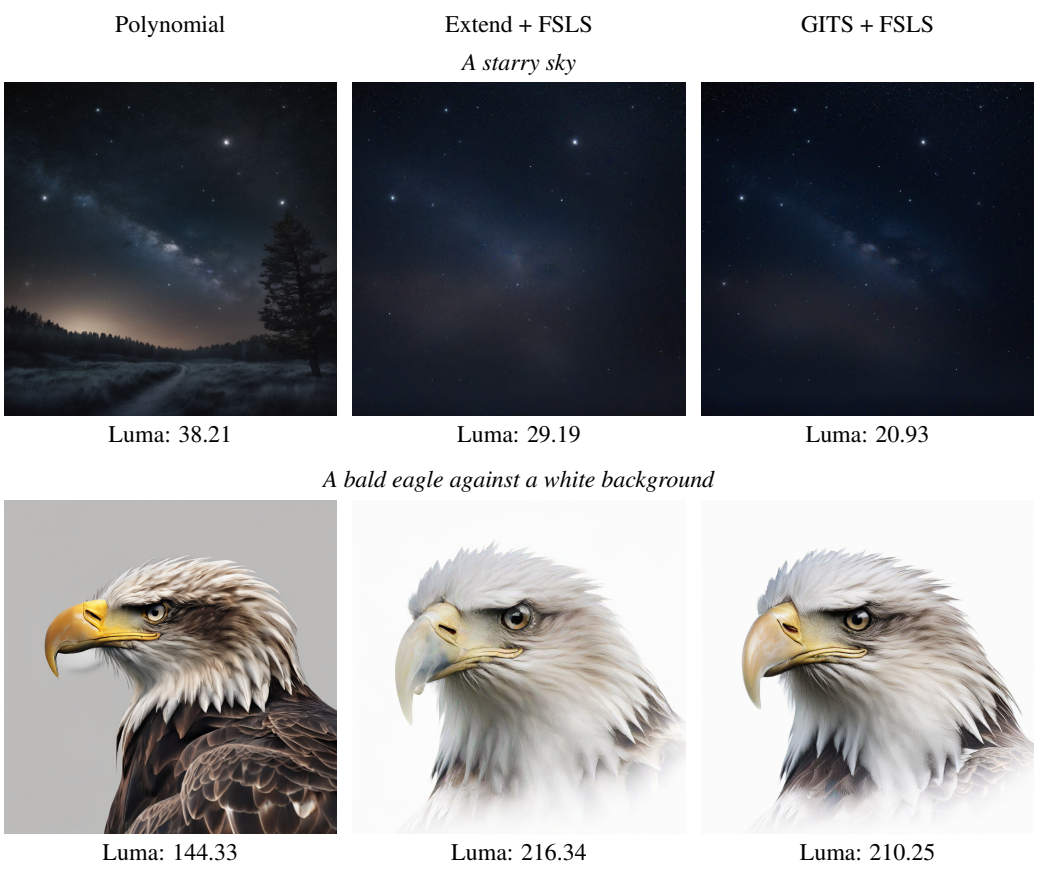


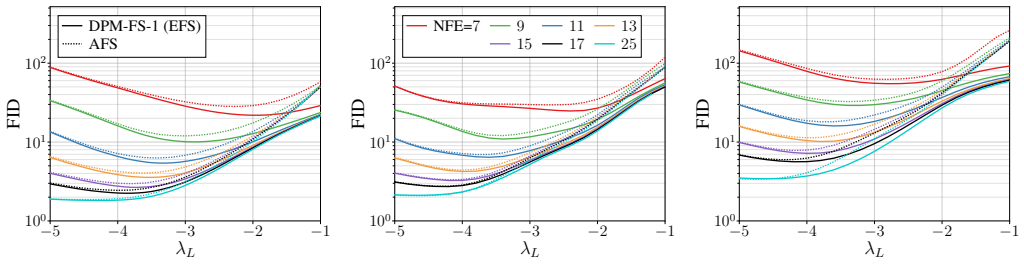
Figure 5: SDXL samples with iPNDM-V at NFE=10, CFG 5, and seed 3407. The two FSLS configurations use the trained text-conditioned EFS. Each row shares its prompt across methods; all six original outputs are shown.

B.4 RELATIONSHIP BETWEEN EFS AND AFS

A related technique is the analytical first step (AFS) introduced in GENIE (Dockhorn et al., 2022). Its purpose is to save one NFE by replacing the learned score in the first solver step with the analytical score of the Gaussian initialization. Accordingly, AFS should be viewed as a local calibration of the first update, rather than a signal estimate derived from the data distribution.

Under the VP parameterization, AFS can be written as

$$\bar{\boldsymbol{x}}_{\lambda_2} = \bar{\boldsymbol{x}}_{\lambda_1} + (e^{-\lambda_2} - e^{-\lambda_1})\boldsymbol{x}_{\lambda_1}, \tag{111}$$



(a) CIFAR-10 (b) AFHQv2 (c) FFHQ

Figure 6: Relationship between EFS and AFS under different left endpoints λ_L. All methods are evaluated by sweeping λ_L on CIFAR-10, AFHQv2, and FFHQ.

where $\bar{\boldsymbol{x}}_\lambda := \boldsymbol{x}_\lambda/\alpha_\lambda$. Assuming $\boldsymbol{x}_{\lambda_1} = \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, the corresponding raw state at the next node becomes

$$\boldsymbol{x}_{\lambda_2}^{\mathrm{AFS}} = \alpha_{\lambda_2} \frac{1 - \sigma_{\lambda_1}}{\alpha_{\lambda_1}} \boldsymbol{\epsilon} + \sigma_{\lambda_2} \boldsymbol{\epsilon}. \tag{112}$$

Equivalently, in the decomposition

$$\boldsymbol{x}_{\lambda_2}^{\mathrm{AFS}} = \alpha_{\lambda_2} \hat{\boldsymbol{x}}^{\mathrm{AFS}} + \sigma_{\lambda_2} \hat{\boldsymbol{\epsilon}}^{\mathrm{AFS}}. \tag{113}$$

AFS corresponds to

$$\hat{\boldsymbol{x}}^{\mathrm{AFS}} = \frac{1 - \sigma_{\lambda_1}}{\alpha_{\lambda_1}} \boldsymbol{\epsilon}, \qquad \hat{\boldsymbol{\epsilon}}^{\mathrm{AFS}} = \boldsymbol{\epsilon}. \tag{114}$$

This form makes the limitation of AFS explicit. Its first-step signal is not inferred from the data distribution, but is only a scaled copy of the initial Gaussian noise. Thus, although AFS can be useful as a local first-step calibration, it does not inject dataset-specific information into the trajectory.

EFS is based on a different endpoint principle. Since the left boundary is naturally regular in signal form, EFS introduces an explicit signal estimate in the first step before the interior denoising dynamics dominate the trajectory. In the simplest unconditional setting, this estimate can be chosen as the dataset mean. In conditioned settings, it can be made condition-dependent. This is also the idea used in Appendix B.2, where a prompt-conditioned estimate is constructed and injected only at the first reverse step.

Figure 6 shows that the two first-step replacements behave similarly in trend but differ consistently in quality. Across most evaluated settings, EFS is comparable to or better than AFS, with clearer advantages on AFHQv2 and FFHQ where the first-step signal must encode richer global structure.

B.5 Relationship to learned first-step correction

OMS (Hu et al., 2024) also introduces prompt-dependent information at the first reverse step while retaining the pretrained sampler thereafter. EFS instead follows from the conditional-mean estimate at the exact low-SNR limit, where the signal anchor does not require the initial noise as an input. DPM-FS additionally supplies explicit higher-order boundary formulas.

B.6 Pseudocode

We summarize the endpoint-aware sampling components used in our experiments. We use a 1-indexed schedule $\lambda_1 < \lambda_2 < \cdots < \lambda_{N-1} < \lambda_N$. Here λ_2 and λ_{N-1} are the transition nodes connected to the finite interior solver, corresponding to λ_L and λ_R in the main text. The outer nodes λ_1 and λ_N are endpoint-side auxiliary nodes used by the second-order closures.

DPM-FS returns the discrete endpoint predictor $\tilde{\boldsymbol{x}}_\theta^{(k)}$ used to construct the first finite state $\boldsymbol{x}_{\lambda_2}$. DPM-LS directly returns the final clean-signal estimate after the last finite state $\boldsymbol{x}_{\lambda_{N-1}}$. The NFE cost of each module is the number of denoising-network evaluations used in the corresponding algorithm, and EFS itself is counted as zero NFE when it is precomputed.

Algorithm 1 DPM-FS

Require: Initial noise $\boldsymbol{\epsilon}$, signal predictor $\boldsymbol{x}_\theta$, first two nodes $\lambda_1 < \lambda_2$, order $k \in \{1, 2\}$

Obtain an initial signal estimate $\hat{\boldsymbol{x}}^{\text{EFS}}$
if $k = 1$ **then**
    return $\hat{\boldsymbol{x}}^{\text{EFS}}$
end if
$\boldsymbol{x}_{\lambda_1} \leftarrow \alpha_{\lambda_1}\hat{\boldsymbol{x}}^{\text{EFS}} + \sigma_{\lambda_1}\boldsymbol{\epsilon}$
$\hat{\boldsymbol{x}}^{(1)} \leftarrow \boldsymbol{x}_\theta(x_{\lambda_1}, \lambda_1)$
$r_{\text{FS}} \leftarrow \dfrac{1}{2}e^{\lambda_2 - \lambda_1}$
return $(1 - r_{\text{FS}})\hat{\boldsymbol{x}}^{\text{EFS}} + r_{\text{FS}}\hat{\boldsymbol{x}}^{(1)}$

Algorithm 2 DPM-LS

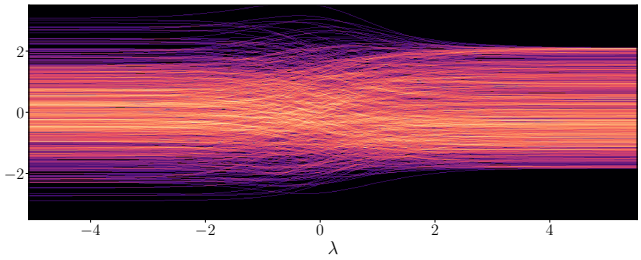
Require: Current latent $\boldsymbol{x}_{\lambda_{N-1}}$, noise predictor $\boldsymbol{\epsilon}_\theta$, last two nodes $\lambda_{N-1} < \lambda_N$, order $k \in \{1, 2\}$

$\hat{\boldsymbol{\epsilon}}^{(0)} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_{N-1}}, \lambda_{N-1})$
$\hat{\boldsymbol{x}}^{(0)} \leftarrow \dfrac{\boldsymbol{x}_{\lambda_{N-1}} - \sigma_{\lambda_{N-1}}\hat{\boldsymbol{\epsilon}}^{(0)}}{\alpha_{\lambda_{N-1}}}$
if $k = 1$ **then**
    return $\hat{\boldsymbol{x}}^{(0)}$
end if
$\boldsymbol{x}_{\lambda_N} \leftarrow \alpha_{\lambda_N}\hat{\boldsymbol{x}}^{(0)} + \sigma_{\lambda_N}\hat{\boldsymbol{\epsilon}}^{(0)}$
$\hat{\boldsymbol{\epsilon}}^{(1)} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\lambda_N}, \lambda_N)$
$r_{\text{LS}} \leftarrow \dfrac{1}{2(1 - e^{\lambda_{N-1} - \lambda_N})}$
$\hat{\boldsymbol{\epsilon}} \leftarrow (1 - r_{\text{LS}})\hat{\boldsymbol{\epsilon}}^{(0)} + r_{\text{LS}}\hat{\boldsymbol{\epsilon}}^{(1)}$
return $\dfrac{\boldsymbol{x}_{\lambda_{N-1}} - \sigma_{\lambda_{N-1}}\hat{\boldsymbol{\epsilon}}}{\alpha_{\lambda_{N-1}}}$

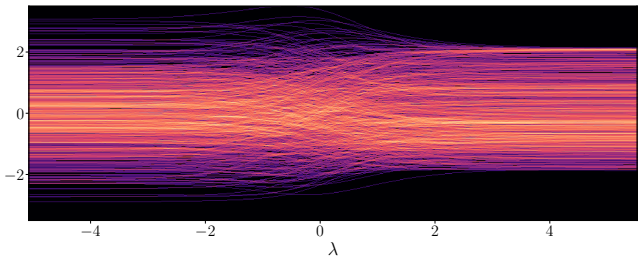
C TRAJECTORY AND PREDICTION DIAGNOSTICS

We inspect the normalized state, recovered signal, and recovered noise along sampled paths. These views separate motion caused by the known mixing coefficients from changes in the quantities approximated by signal- or noise-form solvers. The plots are qualitative coordinate diagnostics; visible variation is not itself an estimate of curvature, prediction error, or truncation error.

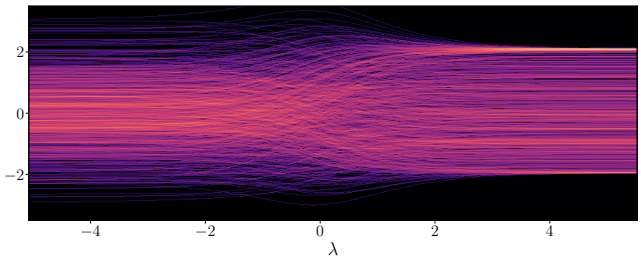
C.1 NORMALIZED-STATE TRAJECTORIES



(a) $\boldsymbol{v}$ predictor



(b) signal predictor



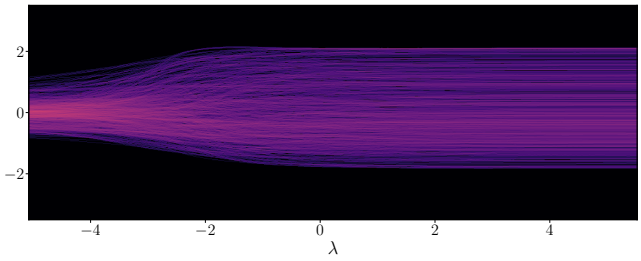
(c) noise predictor

Figure 7: Sampling trajectories of the normalized state $\boldsymbol{x}_\lambda$ for $\boldsymbol{v}$ predictor, signal predictor, and noise predictor. Noise predictor exhibits a stronger concentration near the range boundaries, indicating a more biased transport pattern in normalized state space.

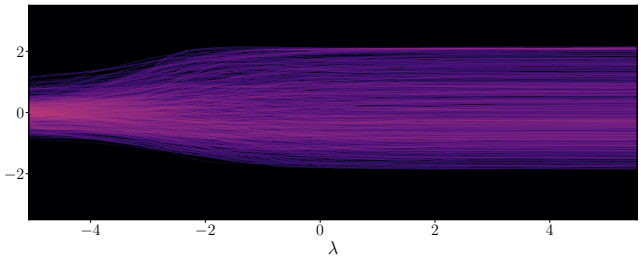
In Figure 7(a), the normalized state changes most visibly in the middle of the log-SNR range and varies less at the extremes. This does not locate the hardest intervals for every numerical solver: even fixed signal and noise directions produce a changing state as α_λ and σ_λ vary. The next two figures inspect the recovered quantities directly.

C.2 RECOVERED-SIGNAL TRAJECTORIES

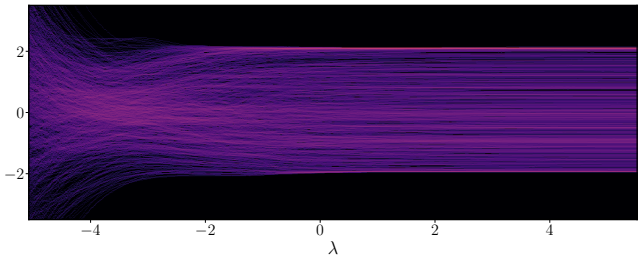
Figure 8 visualizes the recovered signal trajectories. This view makes the left-end asymmetry particularly clear. Near the initial sampling regime, $\boldsymbol{v}$ predictor and signal predictor recover a comparatively coherent coarse signal, whereas noise predictor exhibits a visibly less stable signal estimate. This is exactly the failure mode suggested by Figure 1: when a noise predictor is converted into signal form near the first step, the recovery $\hat{\boldsymbol{x}} = (\boldsymbol{x}_\lambda - \sigma_\lambda \hat{\boldsymbol{\epsilon}})/\alpha_\lambda$ contains the singular factor



(a) $\boldsymbol{v}$ predictor



(b) signal predictor



(c) noise predictor

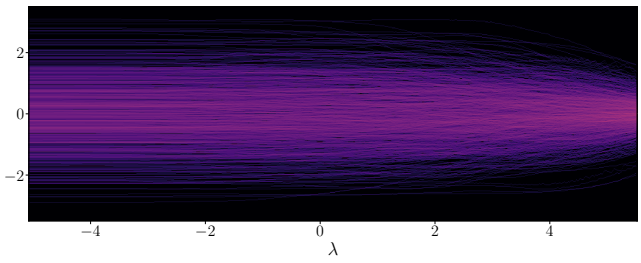
Figure 8: Sampling trajectories of the recovered signal $\boldsymbol{x}$ for $\boldsymbol{v}$ predictor, signal predictor, and noise predictor. Noise predictor is visibly less stable in the initial sampling regime, showing that recovering signal from the noise predictor is inaccurate near the left endpoint.

$1/\alpha_\lambda$. If $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon} + \delta_{\boldsymbol{\epsilon}}$, then

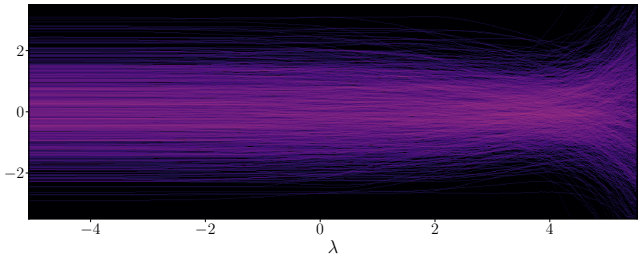
$$\hat{\boldsymbol{x}} - \boldsymbol{x} = -\frac{\sigma_\lambda}{\alpha_\lambda}\delta_{\boldsymbol{\epsilon}} = -e^{-\lambda}\delta_{\boldsymbol{\epsilon}}, \tag{115}$$

so even small noise-prediction errors are amplified as $\lambda \to -\infty$. Because this error occurs at the beginning of sampling, it can affect coarse low-frequency structure and persist through later denoising steps.

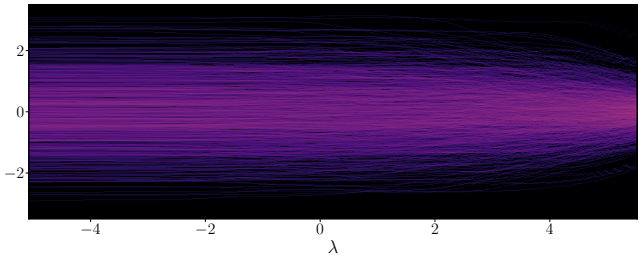
C.3 Recovered-noise trajectories



(a) $\boldsymbol{v}$ predictor



(b) signal predictor



(c) noise predictor

Figure 9: Sampling trajectories of the recovered noise $\boldsymbol{\epsilon}$ for $\boldsymbol{v}$ predictor, signal predictor, and noise predictor. Signal predictor becomes increasingly unstable in the terminal sampling regime, indicating that recovering noise from the signal predictor is inaccurate near the right endpoint.

Figure 9 shows the dual right-end failure. When a signal predictor is converted into noise form, $\hat{\boldsymbol{\epsilon}} = (\boldsymbol{x}_\lambda - \alpha_\lambda \hat{\boldsymbol{x}})/\sigma_\lambda$. If $\hat{\boldsymbol{x}} = \boldsymbol{x} + \delta_{\boldsymbol{x}}$, then

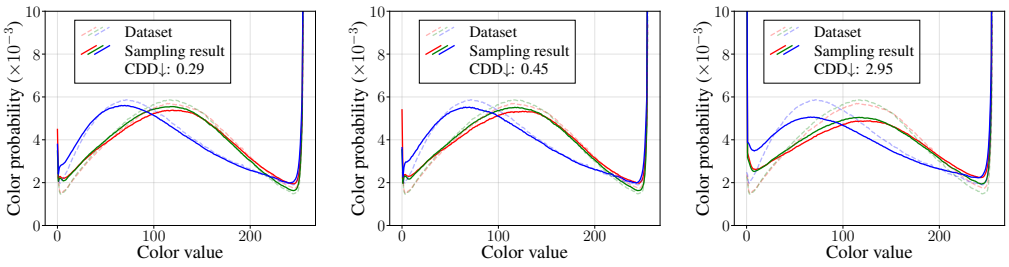
$$\hat{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon} = -\frac{\alpha_\lambda}{\sigma_\lambda}\delta_{\boldsymbol{x}} = -e^{\lambda}\delta_{\boldsymbol{x}}. \tag{116}$$

Thus signal-prediction errors are amplified near $\lambda \to +\infty$, explaining the instability of recovered-noise trajectories near the terminal boundary. This is the right-end counterpart of the left-end signal-recovery failure.

For the velocity model, the recovered signal changes predominantly toward the low-SNR side, while the recovered noise changes toward the high-SNR side. Their more stable middle portions help explain why a simple signal- or noise-form interior update can be competitive after boundary treatment. The unstable cross-target conversions in Figure 8(c) and Figure 9(b) are a separate effect

from this variation along a well-behaved velocity-model trajectory. The conversion identities establish error amplification; they do not by themselves determine the final FID or perceptual impact.

C.4 Color-distribution diagnostics



(a) $\boldsymbol{v}$ predictor (b) Signal predictor (c) Noise predictor

Figure 10: Color-distribution diagnostics on CIFAR-10 for different prediction targets. Dashed curves denote the CIFAR-10 training-set RGB color distributions, and solid curves denote the corresponding distributions of 50,000 generated samples. Each curve reports the marginal probability of observing a given color value in one RGB channel. CDD is the KL-divergence-based color-distribution distance between the dataset distribution and the sampling-result distribution; lower values indicate better color restoration.

To further diagnose whether different prediction targets preserve low-level image statistics, we compare the marginal RGB color distributions of generated CIFAR-10 samples with those of the CIFAR-10 training dataset. Following the same sampling protocol as the FID evaluation, we generate 50,000 images and compare their color statistics against the dataset.

For each color channel $c \in \{R, G, B\}$ and color value $v \in \{0, \dots, 255\}$, let $p_c^{\text{data}}(v)$ and $p_c^{\text{gen}}(v)$ denote the empirical color probabilities of the dataset and the sampling results, respectively. We define the color-distribution distance (CDD) as the KL-divergence-based distance between these two color distributions:

$$\text{CDD} = \frac{1}{3} \sum_{c \in \{R,G,B\}} D_{\text{KL}} \left(p_c^{\text{data}} \,\|\, p_c^{\text{gen}} \right), \tag{117}$$

A lower CDD indicates that the generated samples better reproduce the dataset color distribution. Unlike FID, which measures distributional similarity in a learned feature space, CDD isolates whether the sampler preserves the marginal color and brightness statistics of the target dataset.

Figure 10 shows that both the $\boldsymbol{v}$ predictor and the signal predictor achieve low CDD, indicating better color consistency with the training distribution. In contrast, the noise predictor produces a substantially larger CDD. Consistent with the recovered-signal trajectories in Figure 8, this suggests that the left-end signal-recovery error is propagated into low-level image statistics. The mismatches near the low and high color-value regions correspond to over-dark and over-bright artifacts in image space, providing an additional diagnostic for the low-frequency bias induced by noise-form endpoint mismatch.

D Sampling schedules and endpoint allocation

D.1 Evaluation budgets

Using the N-node construction in Section 3.3, FSLS with precomputed EFS and no FS caching requires $1 + c(N-3) + 2$ field evaluations when each interior update costs c evaluations. Caching and online initialization are included in the actual call count. The main comparisons match total NFE, whereas the transition sweeps in Figure 12 match schedule nodes; the caption specifies the resulting NFE offsets.

D.2 Polynomial, Split, and Extend schedules

We use the Extend construction in Eq. (22) with $\rho = 7$. In the four-dataset experiments, the reference physical-noise range is $[0.002, 80]$; implementing node placement in $(\sigma^{\mathrm{phys}})^{1/\rho}$ is equivalent up to a constant coordinate scale. The Split ablation retains the outer Polynomial nodes as auxiliary anchors while varying the transition under study. Extend instead uses the auxiliary spacing specified in Section 3.4; since h depends on both transitions, varying $\varkappa_L$ also moves the right auxiliary node.

The four-dataset analytic comparisons use $\varkappa_L = 2$, $\varkappa_R = 1.5$, and matched reference-node counts. RAE, PixelDiT, and SDXL use $\varkappa_L = \varkappa_R = 1.5$, with model-specific coordinates detailed in Appendix E.1. For GITS calibration, the image benchmarks use $A = \max(N, 5)$.

D.3 GITS adaptation to endpoint closures

Ordinary GITS follows Chen et al. (2024) with $\gamma = 1.15$. For the image benchmarks, the fourth-order iPNDM teacher uses 61 Polynomial nodes over physical noise $[0.002, 80]$; its 60 interior evaluations plus a terminal clean prediction supply the teacher images for calibration. SDXL uses the native bounds in Appendix E.1.

Node parameterization. For FSLS, we optimize $(\varkappa_L, \varkappa_R, \theta, \delta_{\mathrm{FS}}, \delta_{\mathrm{LS}})$, with main nodes parameterized by $s \in [0, 1]$ as

$$g_\theta(s) = s + 4(\theta - \tfrac{1}{2})s(1 - s), \qquad u(s) = [1 - g_\theta(s)]u_L + g_\theta(s)u_R. \tag{118}$$

The constraint $1/4 < \theta < 3/4$ preserves node ordering; θ is inactive with only two main nodes. Auxiliary nodes satisfy $\lambda_1 = \lambda_L - \delta_{\mathrm{FS}}$ and $\lambda_N = \lambda_R + \delta_{\mathrm{LS}}$, with positive gaps and all nodes inside the fixed noise range.

Calibration and selection. Each candidate runs the requested solver with its endpoint modules and cache setting. Let a_i, b_i be the 2048-dimensional Inception features of student and teacher images generated from the same seed and condition. With $k(a, b) = (a^\top b/2048 + 1)^3$, minimize

$$\widehat{\mathrm{MMD}}^2 = \frac{1}{m(m-1)} \sum_{i \neq j} \left[k(a_i, a_j) + k(b_i, b_j) - k(a_i, b_j) - k(b_i, a_j)\right]. \tag{119}$$

All four diagonal terms are excluded, including paired cross terms. The 256 calibration pairs comprise 128 for selection and 128 for reporting only. Seeds used disjoint from their final evaluation seeds.

Two initial candidates use the Extend main nodes, with either analytic auxiliary gaps or gaps of $\log 4$. Two coordinate rounds evaluate both signs from the current point, using steps $(0.25, 0.25, 0.1, 0.3, 0.3)$ and then half those steps; invalid and duplicate candidates are skipped. This evaluates at most $K = 22$ candidates, or $K = 18$ with two main nodes. The FS-only variant optimizes $(\varkappa_L, \theta, \delta_{\mathrm{FS}})$, fixes the terminal physical noise to 0.002, and uses conventional terminal denoising, with $K = 14$.

For student budget B, calibration uses at most $256(61 + KB)$ field evaluations before teacher reuse, excluding decoding and feature extraction. The schedule bank is fixed before final FID and CLIPScore evaluation.

On EDM2-S with iPNDM-V at 5 NFE and Cache FS enabled for FSLS, offline schedule construction for GITS with FSLS takes approximately $9.8\times$ as long as ordinary GITS (23.74 s versus 2.42 s on $8\times$H100). The resulting schedules can be cached in a schedule bank and reused, avoiding repeated schedule-construction overhead during subsequent sampling.

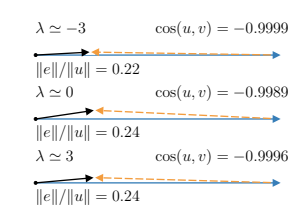
D.4 Prediction trajectories and the role of the interior solver

The trajectory diagnostics can be related directly to the interior integration error. Let $D_\lambda = \boldsymbol{x}_\theta(\boldsymbol{x}_\lambda, \lambda)$ along the model ODE. DDIM holds this prediction fixed at D_i over a step, so Eq. (3) gives the local defect, starting from the same $\boldsymbol{x}_i$, as

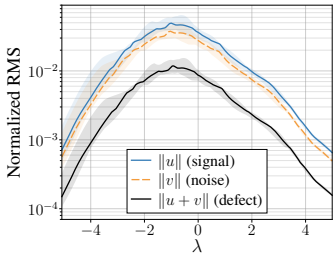
$$\sigma_j \int_{\lambda_i}^{\lambda_j} e^\lambda (D_\lambda - D_i)\, \mathrm{d}\lambda. \tag{120}$$

Thus the changing mixture coefficients need not themselves cause a large discretization error: the relevant quantity is the weighted variation of the prediction over the step. The relatively stable interior predictions in Figure 8 and Figure 9 are consistent with DDIM remaining effective after FSLS handles the boundary regions, even when the state trace in Figure 7 varies visibly.

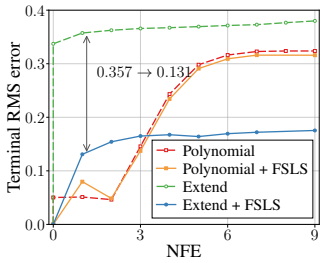
Figure 11 decomposes the DDIM-minus-reference difference as $e = u + v$, with $u = \alpha_j(D_i - D_j^*)$ and $v = \sigma_j(E_i - E_j^*)$, using unit-variance coefficients and reference predictions D_j^*, E_j^* along the same learned ODE. The opposing terms explain why their separate magnitudes overstate the local defect; to assess the accumulated error, panel (c) instead continues each actual sampler state to a common endpoint with the reference solver.



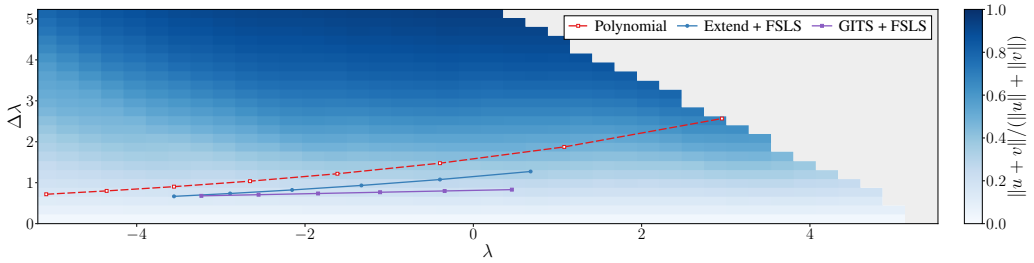
(a) Cancellation geometry



(b) Local defect



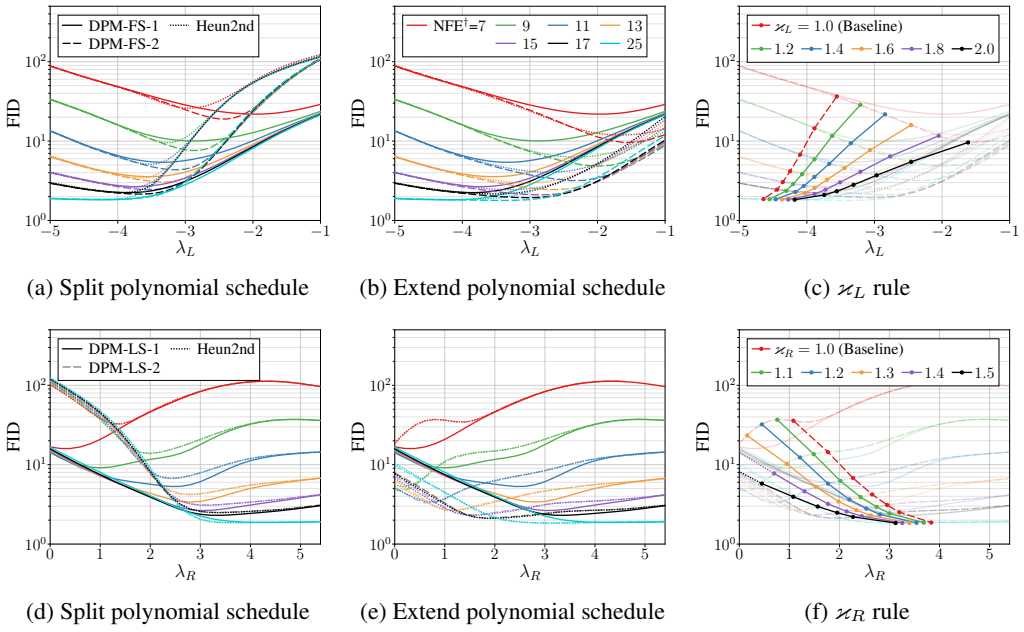
(c) Accumulated error



(d) Residual fraction

Figure 11: Local cancellation and accumulated DDIM error on CIFAR-10. (a,b) The signal and noise terms nearly cancel, leaving a much smaller local defect. (c) Continuing intermediate states to a common endpoint shows that Extend + FSLS finishes with lower error at the same sampling budget. (d) The FSLS schedules place their interior steps in regions of stronger cancellation (lighter colors).

D.5 TRANSITION SENSITIVITY AND MODEL-SPECIFIC ALLOCATION



(a) Split polynomial schedule (b) Extend polynomial schedule (c) $\varkappa_L$ rule

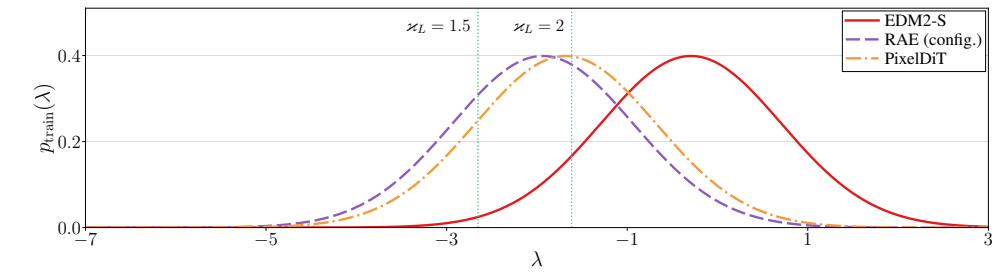
(d) Split polynomial schedule (e) Extend polynomial schedule (f) $\varkappa_R$ rule

Figure 12: Sensitivity of FID to endpoint transition locations on CIFAR-10. The top row evaluates DPM-FS by sweeping the left transition point λ_L, while the bottom row evaluates DPM-LS by sweeping the right transition point λ_R. The first two columns compare split polynomial and extend polynomial schedules. The last column evaluates step-dependent transition rules parameterized by $\varkappa_l$ and $\varkappa_r$ under extend polynomial schedule. Colors encode NFE, and line styles denote solver variants. †The indicated NFE corresponds to DPM-FS/LS-1. DPM-FS/LS-2 uses one additional NFE, and the Heun2nd baseline uses two additional NFEs under the same schedule-node setting.

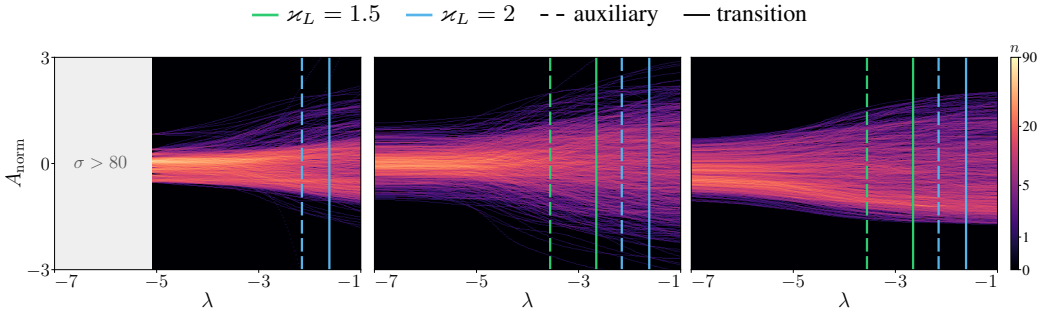
Figure 12 varies the transition locations using both Split and Extend on CIFAR-10. Moving a transition inward assigns more of the trajectory to its boundary closure and leaves a shorter finite interval to the interior solver. Moving it too far inward also increases the boundary remainder scales in Proposition 1: $\mathcal{O}(\alpha_L e^{k\lambda_L})$ on the left and $\mathcal{O}(e^{-(k+1)\lambda_R})$ on the right. The transition is therefore an error-allocation choice, not merely a way of including more extreme noise levels. Because these sweeps match nodes rather than NFE, their order comparisons must be read with the evaluation offsets in the caption.

The CIFAR-10/Heun2nd sweeps motivate the analytic values $\varkappa_L = 2$ and $\varkappa_R = 1.5$ used in the four-dataset comparisons. RAE instead uses $\varkappa_L = \varkappa_R = 1.5$; its GITS+FSLS search starts from these values before optimizing the coordinates. The four-dataset tables support broad usefulness of the fixed analytic choice within those experiments: adding FSLS to Extend improves 141 of 144 uncached dataset–solver–budget comparisons. For RAE, the reported analytic allocation gives uncached FIDs of 9.88–16.65 and improves on GITS for five of the six solvers. The GITS and Extend rankings still vary with the model and solver, as the Heun2nd comparisons in the main text show.

Training-time shifts alter the log-SNR region emphasized during training and can shift where signal structure develops along the learned sampling trajectory. Figure 13 illustrates these model-dependent displacements in shared normalized coordinates, motivating corresponding adjustments to the endpoint transitions.



(a) Training-time distributions



(b) EDM2-S (c) RAE (d) PixelDiT

Figure 13: Training-time shifts and sampling trajectories on ImageNet-512. (a) Normalized log-SNR densities implied by released training configurations. (b–d) Accumulated signal $A_{\text{norm}} = (y - \boldsymbol{\epsilon})e^{-\lambda}$ in the normalized noise-divided state y. Dashed and solid colored lines mark Extend auxiliary and transition nodes, respectively.

E Additional Experiments

E.1 Model and Evaluation Configuration

Training and hardware. We train the five models for CIFAR-10, AFHQv2, and FFHQ on eight NVIDIA RTX 5090 GPUs; all other experiments use eight NVIDIA H100 GPUs. Low-resolution training follows the EDM recipe in Section 4.1, using PyTorch 2.8.0 and CUDA 12.8. CIFAR-10 and AFHQv2 are class-conditioned; FFHQ is unconditional. The conditional AFHQv2 reference is reproduced with the official EDM code.

EDM2-S. We use `edm2-img512-s-2147483-0.130.pkl` with guidance 1, $4 \times 64 \times 64$ latents, and the `sd-vae-ft-mse` decoder. The normalized VE coordinates are $z = x/0.5$ and $\lambda = \log(0.5/\sigma^{\text{phys}})$.

RAE. We use DiTDH-XL (400 epochs) with the DiTDH-S autoguidance model (20 epochs), DINOv2-with-registers-base representations, and the released ViT-XL decoder (Zheng et al., 2026). The $768 \times 32 \times 32$ latent state retains the released normalization, and autoguidance is 1.5 throughout the native time interval. Model evaluation and decoding use FP32; solver arithmetic uses FP64.

PixelDiT. We use `imagenet512_pixeldit_xl_ema.pt` (Yu et al., 2026) in channel-centered, unit-variance pixel coordinates, with $\mu = (-0.0298995, -0.0840310, -0.1853373)$ and $s = 0.5583292255$. These statistics are measured on the ImageNet training set using the EDM2 512^2 crop; s^2 is the mean channel-centered population variance. CFG is 3.5 for native $0.1 < t < 1$ and 1 otherwise; model inputs use FP32, evaluation uses BF16 autocast, and solver arithmetic uses FP64.

ImageNet evaluation. The three backbones share evaluation seeds 0–49999, the per-seed class-label sequence, and the official EDM2 FID reference statistics. RAE and PixelDiT use batch size 4

per GPU; their EFS class means use all ImageNet training images with the EDM2 512^2 center crop and no flips. Both retain the specified schedule nodes without an additional native-sampler time warp.

SDXL and COCO-30k. We use `stabilityai/stable-diffusion-xl-base-1.0`. The COCO val2014 subset contains 30,000 distinct images, one original caption per image. Negative prompts are empty, and sampling uses batch size 4 per GPU with the budget in Section 4.4. The UNet and text encoders use FP16; the VAE and CLIP metric model use FP32; solver arithmetic uses FP642. Latent normalization statistics use only COCO training images.

FID uses `torch-fidelity` Inception-v3 features after directly resizing both real and generated RGB images to 256×256 with bicubic interpolation. CLIPScore uses ViT-L/14 with native preprocessing and 77-token truncation.

E.2 Complete solver and schedule comparisons

Tables 3–6 retain every evaluated Polynomial, Extend, and GITS configuration at NFE 5, 7, 9, 11, 13, and 15. An Endpoint checkmark enables DPM-FS-2 and DPM-LS-2. GITS uses the original search when endpoints are disabled and the endpoint-aware adaptation in Appendix D.3 when they are enabled. For multistep methods, Cache FS inserts the initial signal estimate into the solver history. Gray cells identify Extend/GITS configurations with FSLS. Within each solver and budget, black bold, underlining, and red bold mark the lowest, second-lowest, and highest FID, respectively.

Table 3: Additional FID-50k comparison on CIFAR-10. Lower is better.

Solver	Schedule	Endpoint	Cache FS	NFE 5	7	9	11	13	15
DDIM	Polynomial	–	–	48.56	27.27	17.79	**13.07**	**10.33**	**8.51**
		✓	–	**154.05**	**36.77**	15.57	11.45	9.42	7.99
	Extend polynomial	–	–	24.02	22.16	**21.55**	9.94	7.69	6.55
		✓	–	13.71	12.82	**6.77**	**5.75**	**5.29**	**4.91**
	GITS	–	–	30.07	16.57	11.03	8.38	6.82	6.06
		✓	–	**10.53**	**7.89**	8.39	7.30	6.30	5.88
Heun2nd	Polynomial	–	–	**308.57**	93.48	35.65	14.54	6.88	4.26
		✓	–	100.30	35.15	14.04	6.57	4.11	3.04
	Extend polynomial	–	–	14.98	15.98	16.67	4.58	2.74	2.23
		✓	–	11.31	11.83	4.45	2.61	2.15	2.02
	GITS	–	–	286.16	**178.27**	**87.73**	**51.06**	**41.10**	**21.30**
		✓	–	**8.76**	**3.78**	**2.42**	**2.46**	**1.99**	**1.99**
DPM-Solver++ 2S	Polynomial	–	–	67.97	24.57	14.50	10.05	**7.55**	**5.98**
		✓	–	**176.58**	**83.90**	**31.79**	**11.72**	5.88	4.24
	Extend polynomial	–	–	21.35	19.79	18.76	7.19	5.34	4.46
		✓	–	13.03	12.36	5.61	4.05	3.54	3.25
	GITS	–	–	63.41	25.08	11.60	7.69	6.32	4.92
		✓	–	**8.94**	**6.04**	**4.44**	**3.69**	**3.16**	**2.99**
DPM-Solver++ 2M	Polynomial	–	–	28.56	11.09	6.06	4.17	3.24	2.73
		✓	–	**132.00**	17.59	5.65	4.18	3.44	2.93
		✓	✓	129.59	17.35	5.68	4.22	3.45	2.95
	Extend polynomial	–	–	20.31	**19.29**	**19.00**	**6.87**	**4.54**	**3.49**
		✓	–	13.04	12.33	5.13	3.17	2.57	**2.34**
		✓	✓	12.41	12.16	5.20	3.23	2.60	2.36
	GITS	–	–	14.58	7.65	4.32	3.06	2.56	2.53
		✓	–	**8.35**	**4.40**	3.61	3.20	**2.52**	2.46
		✓	✓	8.37	4.49	**3.42**	**2.89**	2.69	2.47
UniPC	Polynomial	–	–	28.57	10.57	5.51	3.65	2.80	2.38
		✓	–	**131.29**	16.35	5.08	3.71	3.01	2.57
		✓	✓	130.10	16.43	5.06	3.67	2.97	2.53
	Extend polynomial	–	–	19.61	**18.79**	**18.57**	**6.82**	**4.47**	**3.41**
		✓	–	12.79	12.21	4.95	2.95	2.36	2.15
		✓	✓	13.59	14.38	5.04	2.95	2.33	2.12
	GITS	–	–	13.95	7.59	4.18	2.59	2.16	2.09
		✓	–	8.27	**3.74**	**2.78**	2.37	2.15	**2.07**
		✓	✓	**4.68**	4.09	2.80	**2.34**	**2.13**	2.08
iPNDM-V	Polynomial	–	–	28.78	11.46	6.60	4.69	3.75	3.19
		✓	–	**150.80**	**19.28**	5.45	4.12	3.54	3.12
		✓	✓	136.58	17.74	5.46	4.20	3.62	3.18
	Extend polynomial	–	–	20.27	17.06	**17.77**	**6.56**	**4.38**	**3.36**
		✓	–	13.31	12.35	5.11	3.10	2.52	2.34
		✓	✓	12.77	11.28	4.69	2.79	**2.23**	**2.04**
	GITS	–	–	10.23	5.70	3.33	2.65	2.35	2.40
		✓	–	9.54	4.83	3.24	2.66	2.36	2.19
		✓	✓	**8.65**	**3.93**	**3.08**	**2.54**	2.28	2.12

Table 4: Additional FID-50k comparison on AFHQv2. Lower is better.

Solver	Schedule	Endpoint	Cache FS	NFE 5	7	9	11	13	15
DDIM	Polynomial	–	–	23.61	14.53	10.26	8.00	**6.65**	**5.75**
		✓	–	**83.90**	22.13	10.80	7.73	6.46	5.66
	Extend polynomial	–	–	22.78	**22.48**	**22.36**	**8.66**	6.09	5.08
		✓	–	13.59	13.45	**4.89**	**3.89**	**3.71**	**3.64**
	GITS	–	–	19.70	12.05	8.64	6.88	5.78	5.34
		✓	–	**7.86**	**4.76**	5.01	4.95	4.59	4.26
Heun2nd	Polynomial	–	–	243.41	54.50	26.52	11.87	6.61	4.21
		✓	–	51.82	23.21	10.80	6.22	4.05	3.15
	Extend polynomial	–	–	18.86	19.50	20.08	5.20	3.44	2.81
		✓	–	12.92	13.14	4.42	**3.17**	2.72	2.48
	GITS	–	–	**305.35**	**208.90**	**54.48**	**26.85**	**16.49**	**9.68**
		✓	–	**7.78**	**5.30**	**3.73**	3.32	**2.31**	**2.15**
DPM-Solver++ 2S	Polynomial	–	–	24.47	13.93	8.75	6.46	**5.20**	**4.39**
		✓	–	**127.60**	**36.25**	16.04	**8.23**	5.03	3.85
	Extend polynomial	–	–	22.03	21.30	**21.06**	6.62	4.30	3.60
		✓	–	13.49	13.34	4.56	3.27	2.96	2.85
	GITS	–	–	29.14	13.74	9.00	6.55	5.09	3.99
		✓	–	**7.56**	**7.06**	**3.57**	**3.19**	**2.84**	**2.77**
DPM-Solver++ 2M	Polynomial	–	–	12.24	6.87	4.51	3.41	2.87	2.57
		✓	–	**62.55**	11.31	5.19	3.74	3.14	2.79
		✓	✓	60.14	11.16	5.24	3.80	3.17	2.81
	Extend polynomial	–	–	21.87	**21.58**	**21.47**	**6.88**	**4.14**	**3.18**
		✓	–	13.50	13.35	4.45	2.96	2.56	2.41
		✓	✓	12.73	13.11	4.31	2.89	**2.55**	2.41
	GITS	–	–	13.50	6.21	4.17	3.60	3.07	2.95
		✓	–	7.19	3.65	2.91	**2.68**	2.58	**2.36**
		✓	✓	**6.61**	**3.53**	**2.74**	2.69	2.57	2.43
UniPC	Polynomial	–	–	12.05	6.69	4.28	3.17	2.66	2.40
		✓	–	**62.00**	10.73	4.86	3.47	2.88	2.57
		✓	✓	60.29	10.73	4.77	3.33	2.75	2.46
	Extend polynomial	–	–	21.59	**21.33**	**21.24**	**6.58**	**3.88**	**3.00**
		✓	–	13.47	13.31	4.39	2.87	2.45	2.31
		✓	✓	13.50	14.50	4.21	2.70	2.35	2.23
	GITS	–	–	12.26	5.59	3.84	3.28	2.62	2.39
		✓	–	5.28	3.50	2.70	2.40	**2.25**	**2.19**
		✓	✓	**5.27**	**3.08**	**2.65**	**2.34**	2.30	2.31
iPNDM-V	Polynomial	–	–	15.55	7.15	4.71	3.69	3.18	2.87
		✓	–	**80.21**	12.42	4.97	3.65	3.19	2.93
		✓	✓	68.54	11.03	5.02	3.71	3.26	2.98
	Extend polynomial	–	–	21.80	**20.50**	**20.81**	**6.84**	**4.15**	**3.14**
		✓	–	13.53	13.36	4.50	3.04	2.62	2.49
		✓	✓	13.36	12.80	4.21	2.73	2.32	**2.17**
	GITS	–	–	9.85	5.38	3.92	3.26	2.86	2.83
		✓	–	7.68	3.73	2.88	2.60	2.40	2.35
		✓	✓	**5.37**	**3.38**	**2.83**	**2.46**	**2.29**	2.18

Table 5: Additional FID-50k comparison on FFHQ. Lower is better.

Solver	Schedule	Endpoint	Cache FS	NFE 5	7	9	11	13	15
DDIM	Polynomial	–	–	58.67	35.68	24.47	18.41	14.63	12.11
		✓	–	**149.05**	**64.14**	29.91	**19.89**	**15.31**	**12.61**
	Extend polynomial	–	–	35.93	31.90	**30.49**	12.39	9.46	8.16
		✓	–	21.35	19.40	**8.98**	**7.83**	**7.93**	**7.71**
	GITS	–	–	42.01	23.93	16.68	12.83	10.48	9.73
		✓	–	**13.58**	**10.31**	12.22	12.05	10.73	9.77
Heun2nd	Polynomial	–	–	346.50	153.06	61.56	31.41	16.72	10.39
		✓	–	166.16	61.31	30.95	16.28	10.00	6.83
	Extend polynomial	–	–	19.34	19.64	21.92	7.25	4.84	4.21
		✓	–	16.90	17.27	7.61	4.61	3.83	3.52
	GITS	–	–	**390.95**	**259.47**	**135.57**	**80.21**	**46.82**	**25.60**
		✓	–	**14.10**	**13.48**	**5.65**	**4.02**	**3.46**	**3.23**
DPM-Solver++ 2S	Polynomial	–	–	65.39	32.76	20.09	13.95	10.39	8.23
		✓	–	**207.03**	**96.85**	**51.72**	**24.36**	**12.90**	**8.45**
	Extend polynomial	–	–	30.95	28.13	25.59	9.28	6.60	5.56
		✓	–	20.12	18.42	7.75	**5.59**	**5.20**	**4.91**
	GITS	–	–	73.21	31.26	16.55	11.09	8.57	6.62
		✓	–	**12.21**	**10.38**	**7.34**	6.35	5.30	4.91
DPM-Solver++ 2M	Polynomial	–	–	30.12	13.82	8.14	5.70	4.46	3.77
		✓	–	**135.52**	**32.98**	10.80	6.83	5.34	4.48
		✓	✓	127.20	31.56	10.81	6.88	5.37	4.50
	Extend polynomial	–	–	28.26	25.53	**24.81**	**8.85**	**6.11**	**4.67**
		✓	–	20.16	18.28	7.35	4.55	3.83	**3.52**
		✓	✓	18.36	17.79	7.32	4.52	**3.82**	**3.52**
	GITS	–	–	17.74	10.30	6.04	**4.35**	4.22	4.14
		✓	–	11.53	**6.50**	6.44	5.09	4.21	3.92
		✓	✓	**11.26**	6.55	**5.45**	4.70	4.40	3.94
UniPC	Polynomial	–	–	29.62	13.07	7.45	5.10	3.94	3.33
		✓	–	**135.09**	**30.38**	9.48	6.03	4.72	3.94
		✓	✓	125.88	29.85	8.97	5.38	4.15	3.47
	Extend polynomial	–	–	26.69	24.49	**24.06**	**9.13**	**6.29**	**4.72**
		✓	–	19.73	17.92	7.30	4.31	3.52	3.23
		✓	✓	17.05	17.29	5.87	3.68	3.20	**2.96**
	GITS	–	–	16.74	10.53	6.05	3.79	3.55	3.30
		✓	–	11.33	6.78	4.87	4.33	3.69	3.57
		✓	✓	**8.91**	**5.18**	**4.01**	**3.54**	**3.18**	3.01
iPNDM-V	Polynomial	–	–	37.18	15.71	9.28	6.61	5.28	4.50
		✓	–	**146.98**	**39.44**	11.56	6.99	5.61	4.87
		✓	✓	134.89	34.14	11.24	7.10	5.75	**5.00**
	Extend polynomial	–	–	27.92	24.68	**23.98**	**8.57**	**5.95**	4.56
		✓	–	20.65	18.20	7.60	4.65	3.91	3.65
		✓	✓	19.18	16.04	6.72	**3.88**	**3.12**	**2.82**
	GITS	–	–	15.93	8.39	5.20	4.03	3.61	4.04
		✓	–	12.29	7.16	5.99	4.80	3.86	3.61
		✓	✓	**11.98**	**6.21**	**4.85**	4.11	3.43	2.95

Table 6: Additional FID-50k comparison on ImageNet-512 (EDM2-S). Lower is better.

Solver	Schedule	Endpoint	Cache FS	NFE 5	7	9	11	13	15
DDIM	Polynomial	–	–	83.99	47.38	27.76	**18.09**	**12.68**	**9.50**
		✓	–	**126.93**	**51.36**	**28.09**	17.98	12.52	9.35
	Extend polynomial	–	–	30.16	22.58	20.35	6.99	**5.36**	**4.78**
		✓	–	8.69	**6.68**	**5.29**	**6.18**	5.86	5.25
	GITS	–	–	54.83	23.48	17.55	13.75	9.61	8.16
		✓	–	**7.94**	8.22	10.97	9.33	8.20	6.58
Heun2nd	Polynomial	–	–	208.43	105.55	47.20	24.77	11.57	7.08
		✓	–	142.79	54.55	25.97	11.66	7.06	5.09
	Extend polynomial	–	–	15.43	13.84	11.93	4.56	4.28	3.88
		✓	–	**7.41**	6.06	4.05	3.94	3.65	3.47
	GITS	–	–	**251.25**	**220.09**	**125.20**	**84.24**	**64.57**	**72.12**
		✓	–	8.54	**5.45**	**3.58**	**3.27**	**3.18**	**3.08**
DPM-Solver++ 2S	Polynomial	–	–	92.96	37.94	**21.68**	**12.33**	**7.60**	**5.60**
		✓	–	**147.13**	**47.61**	17.26	9.01	6.13	4.71
	Extend polynomial	–	–	22.13	17.22	13.98	4.57	3.89	3.56
		✓	–	7.09	6.05	**3.66**	3.89	3.72	3.45
	GITS	–	–	104.78	40.54	9.50	5.44	4.47	4.90
		✓	–	**6.44**	**5.09**	3.66	**3.40**	**3.16**	**3.12**
DPM-Solver++ 2M	Polynomial	–	–	49.18	12.22	4.98	3.63	3.21	2.99
		✓	–	**98.90**	20.56	6.22	3.75	3.21	3.00
		✓	✓	89.29	19.17	6.10	3.75	3.22	3.01
	Extend polynomial	–	–	17.88	15.00	**14.66**	4.48	3.33	2.90
		✓	–	7.10	5.99	3.07	2.92	**2.83**	**2.80**
		✓	✓	6.43	5.86	**3.04**	**2.92**	2.85	2.81
	GITS	–	–	33.63	**29.36**	13.26	**10.93**	**6.86**	**5.62**
		✓	–	6.15	4.18	4.50	3.23	3.09	2.98
		✓	✓	**5.65**	**4.10**	3.95	3.28	3.06	2.90
UniPC	Polynomial	–	–	48.31	10.66	4.38	3.45	3.14	2.94
		✓	–	**98.15**	18.51	5.15	3.38	3.08	2.94
		✓	✓	87.87	17.68	5.13	3.34	3.02	2.88
	Extend polynomial	–	–	15.85	13.82	**13.64**	4.58	3.42	2.89
		✓	–	6.64	5.91	2.94	2.79	2.75	2.74
		✓	✓	6.54	6.69	**2.88**	**2.70**	**2.63**	**2.61**
	GITS	–	–	30.11	**32.55**	11.82	**8.73**	**5.28**	**4.25**
		✓	–	**5.93**	**3.49**	3.23	3.06	2.97	2.95
		✓	✓	7.31	3.56	3.18	2.95	2.83	2.69
iPNDM-V	Polynomial	–	–	60.65	18.28	6.82	4.12	3.27	2.95
		✓	–	**122.32**	**23.01**	7.08	4.15	3.34	3.03
		✓	✓	100.29	20.77	6.92	4.16	3.37	3.05
	Extend polynomial	–	–	17.99	14.54	**14.69**	4.21	3.00	**2.68**
		✓	–	7.70	6.07	3.04	2.80	2.76	2.75
		✓	✓	6.48	5.64	**2.94**	**2.78**	**2.75**	2.72
	GITS	–	–	13.03	15.62	7.38	**6.56**	**5.04**	**4.59**
		✓	–	6.85	4.23	3.87	3.24	2.99	2.98
		✓	✓	**5.87**	**3.61**	3.63	3.31	2.95	2.86

E.3 RAE AND PIXELDIT AT FIVE FIELD EVALUATIONS

Tables 7 and 8 report five-NFE sampling results, using the same Endpoint and Cache FS notation as Tables 3–6. Dashes in the FID columns indicate configurations not reported here. Gray cells mark rows with endpoint closures enabled; bold marks the lowest FID for each solver across the displayed configurations.

Table 7: FID-50k on ImageNet-512 with RAE at NFE=5. Extend uses $\varkappa_L = \varkappa_R = 1.5$. Lower is better.

Solver	Endpoint	Cache FS	Polynomial	Extend Polynomial	GITS
DDIM	–	–	30.06	–	49.36
	✓	–	–	**9.88**	10.02
Heun2nd	–	–	239.09	–	236.86
	✓	–	–	16.65	**10.43**
DPM-Solver++ 2S	–	–	29.51	–	26.02
	✓	–	–	**10.12**	11.30
DPM-Solver++ 2M	–	–	24.70	–	53.17
	✓	–	–	10.10	11.42
	✓	✓	–	**8.82**	10.03
UniPC	–	–	24.59	–	52.24
	✓	–	–	10.18	11.58
	✓	✓	–	**9.81**	11.05
iPNDM-V	–	–	29.02	–	51.87
	✓	–	–	**9.92**	11.26
	✓	✓	–	9.98	10.99

Table 8: FID-50k on ImageNet-512 with PixelDiT at NFE=5, using measured pixel normalization. Extend uses $\varkappa_L = \varkappa_R = 1.5$. Lower is better.

Solver	Endpoint	Cache FS	Polynomial	Extend Polynomial	GITS
DDIM	–	–	33.63	–	19.58
	✓	–	–	14.02	**9.39**
Heun2nd	–	–	200.72	–	218.86
	✓	–	–	64.87	**16.60**
DPM-Solver++ 2S	–	–	35.95	–	36.45
	✓	–	–	13.43	**8.92**
DPM-Solver++ 2M	–	–	28.70	–	20.73
	✓	–	–	13.69	8.85
	✓	✓	–	11.83	**7.95**
UniPC	–	–	28.76	–	20.87
	✓	–	–	14.20	8.89
	✓	✓	–	12.21	**7.36**
iPNDM-V	–	–	31.87	–	19.88
	✓	–	–	13.45	8.95
	✓	✓	–	14.44	**8.77**

E.4 JOINT FIRST- AND LAST-STEP CLOSURE

In Figure 14, FSLS generally improves on first-step correction alone at the same total NFE, despite allocating fewer evaluations to the interior solver. This supports the value of controlling high-SNR termination error after correcting low-SNR initialization. The results therefore favor allocating the limited sampling budget jointly to both endpoint closures and the interior, with a schedule calibrated for the complete sampler.

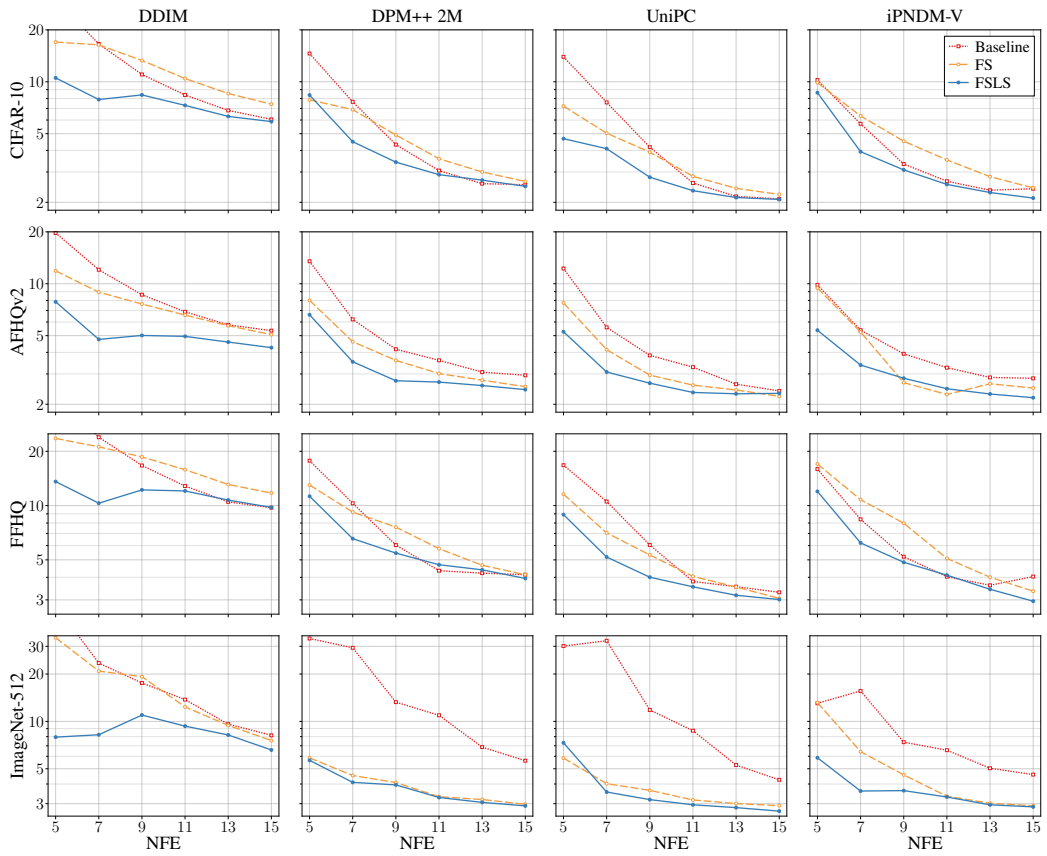


Figure 14: FID-50k for ordinary GITS (Baseline), GITS+FS (FS), and GITS+FSLS (FSLS). Each configuration uses a separately calibrated schedule; multistep endpoint variants enable Cache FS. Each row shares a logarithmic FID axis; curves outside the displayed range are clipped.

E.5 Consistency of the training formulation

We train the variance-normalized velocity model with the EDM architecture and hyperparameters, applying the log-SNR shift in Appendix A.9. Evaluation uses NFE=35 for CIFAR-10 and AFHQv2, and NFE=79 for FFHQ. In Figure 15, solid curves show the lowest FID among three sampling-seed groups; shading shows the other two, rather than a confidence interval.
The curves remain close to the EDM references under this shared training recipe, consistent with the residual-label relation in Section 3.2. This empirical agreement concerns sampling quality and does not establish that the learned predictors are identical.

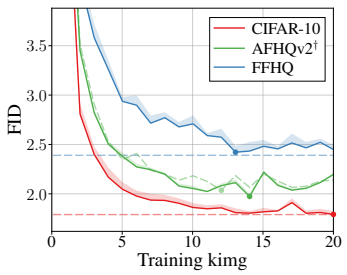


Figure 15: FID during variance-normalized velocity training. Dashed lines: EDM references. †The AFHQv2 reference is our conditional EDM reproduction.

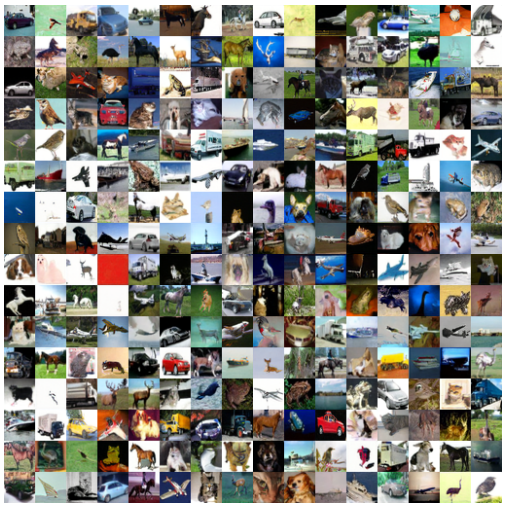
F Additional samples

We first show different predictor combinations on CIFAR-10, then compare iPNDM-V with and without endpoint closures and GITS at NFE=5 on CIFAR-10, AFHQv2, FFHQ, and ImageNet-512.

The final comparison includes RAE and PixelDiT on ImageNet-512. The sample grids use fixed seeds and are shown without cherry-picking.



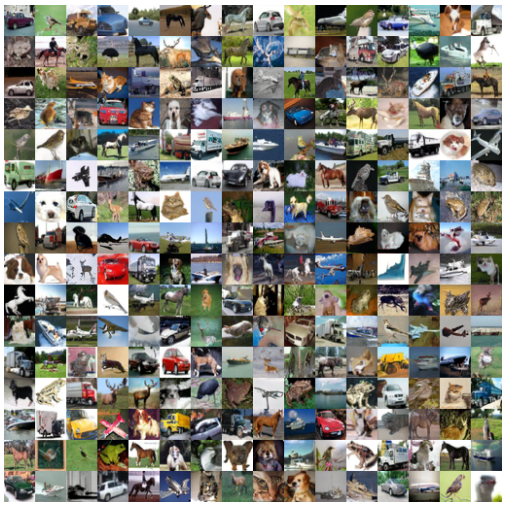
(a) $\epsilon_\theta + \epsilon_\theta$, FID 4.41



(b) $\epsilon_\theta + x_\theta$, FID **5.68**

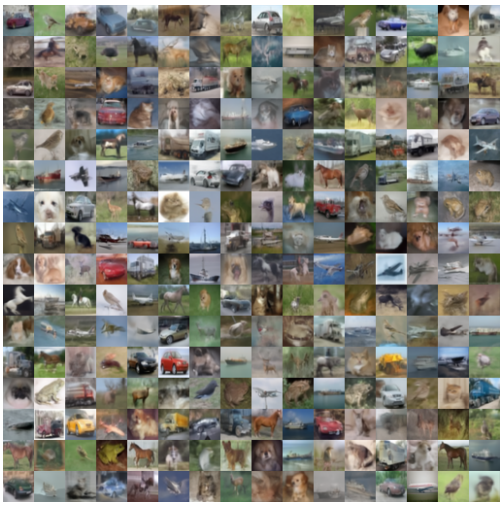


(c) $x_\theta + \epsilon_\theta$, FID **2.11**

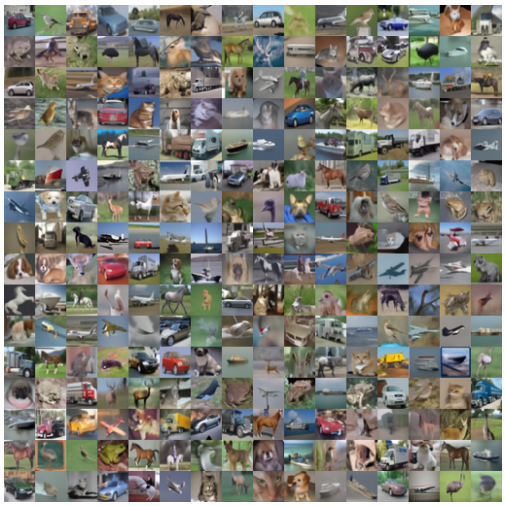


(d) $x_\theta + x_\theta$, FID 3.04

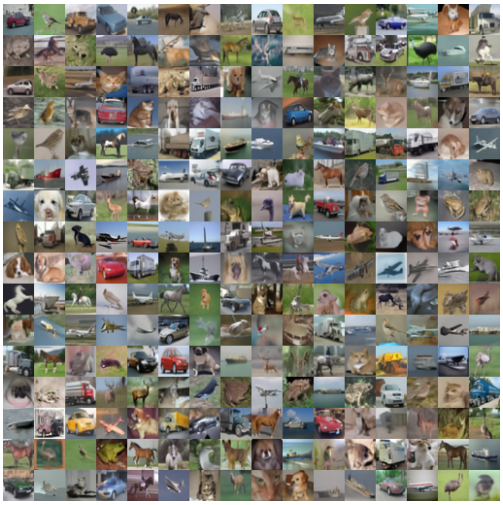
Figure 16: Comparison of different predictor combinations on CIFAR-10. The notation $\epsilon_\theta + \epsilon_\theta$, $\epsilon_\theta + x_\theta$, $x_\theta + \epsilon_\theta$, and $x_\theta + x_\theta$ indicates the predictor used in the early and late stages of sampling, respectively. The FID of each setting is reported below the corresponding sample grid. The lowest FID is highlighted in **bold**, and the highest FID is highlighted in red bold.



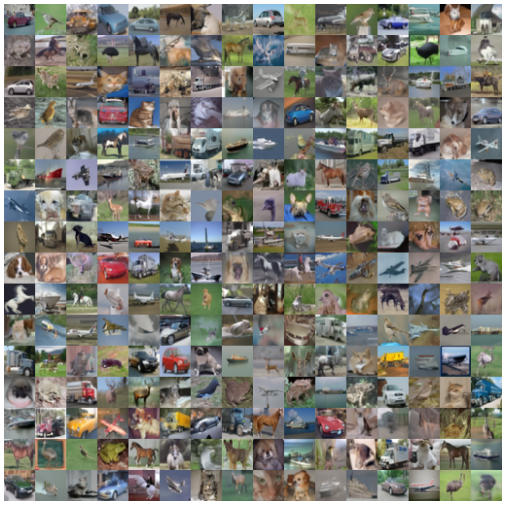
(a) iPNDM, NFE 5, FID 28.78



(b) w/ our methods, NFE 5, FID **12.77**



(c) w/ GITS, NFE 5, FID 10.23



(d) w/ our methods and GITS, NFE 5, FID **8.65**

Figure 17: Additional samples on CIFAR-10.



(a) iPNDM, NFE 5, FID 15.55



(b) w/ our methods, NFE 5, FID **13.36**



(c) w/ GITS, NFE 5, FID 9.85



(d) w/ our methods and GITS, NFE 5, FID **5.37**

Figure 18: Additional samples on AFHQv2.



(a) iPNDM, NFE 5, FID 37.18



(b) w/ our methods, NFE 5, FID **19.18**



(c) w/ GITS, NFE 5, FID 15.93



(d) w/ our methods and GITS, NFE 5, FID **11.98**

Figure 19: Additional samples on FFHQ.



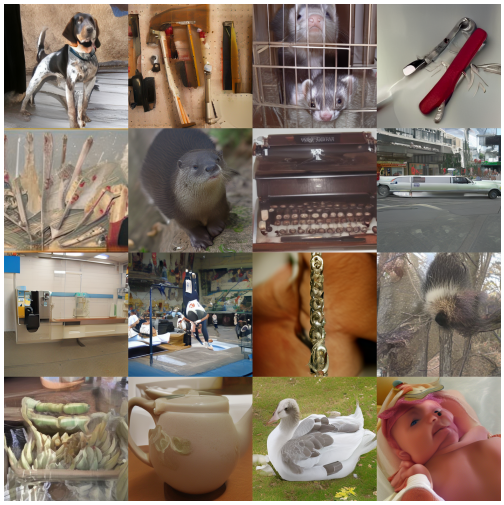
(a) iPNDM, NFE 5, FID 60.65



(b) w/ our methods, NFE 5, FID **6.48**

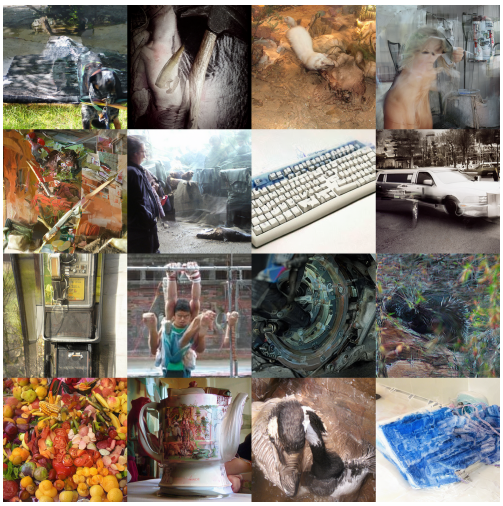


(c) w/ GITS, NFE 5, FID 13.03



(d) w/ our methods and GITS, NFE 5, FID **5.87**

Figure 20: Additional samples on ImageNet-512.



(a) RAE, Polynomial, FID 29.02



(b) RAE, GITS+FSLS, FID **10.99**



(c) PixelDiT, Polynomial, FID 31.87



(d) PixelDiT, GITS+FSLS, FID **8.77**

Figure 21: Additional samples on ImageNet-512.