RESIDUAL-GUIDED SAMPLING: STRUCTURED ACCELERATION AND TIMESTEP SCHEDULING

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient sampling from diffusion and flow models depends jointly on numerical updates and timestep schedules, yet these components are often designed separately. We introduce residual-guided sampling to unify them through a structural residual. For smooth linear data-noise interpolants, pathwise forward acceleration lies in the state-velocity span under a nondegeneracy condition. For the variance-preserving cosine interpolant, augmenting Euler with a second-order acceleration term whose state and velocity coefficients are derived from the forward interpolant recovers deterministic DDIM through second order. However, reverse-trajectory acceleration need not lie in this span, so we approximate it as a weighted sum of state and velocity. With the pretrained velocity field fixed, a lightweight head predicts the two coefficients from time, conditioning, and guidance scale. The resulting acceleration mismatch defines a computable residual. Our analysis shows that it characterizes the leading discretization defect and enters a global error bound; it also guides coefficient fitting and a propagation-aware timestep schedule. The sampler uses the predicted acceleration without extra velocity-field evaluations, while the schedule transfers to other integrators. A single head supports unconditional and conditional sampling with or without classifier-free guidance. Across tested settings, our method improves upon matched Euler, Heun, AB2, and RF-Solver baselines, with the largest gains at low-to-intermediate evaluation budgets.

1 INTRODUCTION

Diffusion and flow-based generative models produce samples by transporting a reference distribution to the data distribution through a stochastic process or a learned vector field. Generation is sequential, with each sampling step requiring one or more network evaluations. The practical objective is therefore to obtain high-quality samples with as few evaluations as possible. Under a fixed evaluation budget, few-step sample quality depends on how each step is computed (the numerical update rule) and where the steps are placed (the timestep schedule). This view encompasses denoising diffusion models, score-based SDEs and probability-flow ODEs, deterministic diffusion trajectories, flow matching, rectified flow, and stochastic interpolants (Ho et al., 2020; Song et al., 2021b;a; Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023; Albergo et al., 2025), including transformer implementations such as SiT (Ma et al., 2024).

Existing methods typically design numerical corrections and timestep schedules separately, using different criteria. Diffusion solvers use non-Markovian paths, semilinear structure, multistep formulas, exponential integration, predictor-corrector updates, and model statistics (Karras et al., 2022; Liu et al., 2022; Zhang & Chen, 2023; Lu et al., 2022; 2025; Zhao et al., 2023; Zheng et al., 2023). Learned and geometry-aware methods distill derivatives, optimize solver parameters, or fit trajectory corrections (Dockhorn et al., 2022; Watson et al., 2022; Zhou et al., 2024; Shaul et al., 2024a;b; Frankel et al., 2025; Wang et al., 2025a; Liu & Yue, 2025; Wang et al., 2026), while scheduling methods decide where evaluations occur (Sabour et al., 2024; Xue et al., 2024; Chen et al., 2024a;b; Tong et al., 2025; Kong et al., 2026; Yuan et al., 2026; Zhou et al., 2026; Jo & Choi, 2026; Gupta et al., 2026). To unify these designs, we introduce *residual-guided sampling*, which uses one computable structural residual for coefficient fitting, error analysis, and fixed-budget timestep selection.

For few-step integration of rectified-flow ODEs (Liu et al., 2023), *reverse acceleration*, the change in velocity along the reverse sampling trajectory, determines the leading local truncation error of Euler

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.

sampling. Motivated by the observation that, under nondegenerate data-to-noise parameterizations, the *forward acceleration* lies in the span of the state and velocity, we approximate the reverse acceleration by a learned linear combination of the current state and velocity and use it as a second-order correction. The state-velocity span is an inductive bias rather than an exact constraint. Under the VP-cosine interpolant, the update matches deterministic DDIM (Song et al., 2021a) to second order.

We use offline Jacobian-vector-product (JVP) targets from the frozen velocity model to train a coefficient head that predicts the two scalar coefficients multiplying state and velocity in the reverse-acceleration approximation. The resulting acceleration mismatch defines a *structural residual* that determines the leading local defect. Our global error bound separates velocity-model and Taylor errors from a propagation-weighted residual term reduced through coefficient fitting and fixed-budget timestep selection. We apply the resulting timestep grid to other integrators to evaluate transfer beyond our acceleration-corrected update. For classifier-free guidance (CFG) (Ho & Salimans, 2021), we treat the conditional-unconditional combination as one guided field and condition the coefficient head on embeddings of time, the conditioning information, and the guidance scale. Thus, the same residual governs both coefficient fitting and timestep allocation for unguided and guided sampling.

2 Related Work

DDPMs reverse a noising process (Ho et al., 2020); score-based modeling connects reverse-time SDEs to probability-flow ODEs (Song et al., 2021b); and DDIM defines deterministic non-Markovian trajectories (Song et al., 2021a). Flow matching and rectified flow regress transport velocities (Lipman et al., 2023; Liu et al., 2023), while stochastic interpolants cover deterministic and stochastic bridge constructions (Albergo & Vanden-Eijnden, 2023; Albergo et al., 2025). SiT implements this framework with transformers (Ma et al., 2024). Together, these developments recast generative modeling as learning and numerically integrating continuous differential equations.

Numerical solvers exploit ODE structure in their update rules. PNDM uses pseudo-numerical and multistep formulas (Liu et al., 2022); DEIS uses exponential integration (Zhang & Chen, 2023); and DPM-Solver analytically integrates part of the diffusion ODE, with DPM-Solver++ extending the formulation to guided sampling (Lu et al., 2022; 2025). UniPC uses predictor-corrector updates (Zhao et al., 2023), DPM-Solver-v3 uses empirical model statistics (Zheng et al., 2023), and EDM studies parameterization, preconditioning, schedules, and solvers (Karras et al., 2022). RF-Solver applies higher-order Taylor approximations to rectified-flow dynamics (Wang et al., 2025b), while A^2S adds an acceleration correction for few-step rectified flow (Fu et al., 2025). Our update learns coefficients in the state-velocity span and retains the unrepresented acceleration as a residual for error analysis and timestep scheduling.

Other methods learn or distill higher-order information from a pretrained first-order model. GENIE distills JVP targets into a full-dimensional prediction for truncated-Taylor sampling (Dockhorn et al., 2022), and FlowTurbo trains a refiner that reuses velocity predictions across flow steps (Zhao et al., 2024). DDSS differentiates through a sample-quality objective (Watson et al., 2022); Bespoke and Bespoke Non-Stationary Solvers learn model-specific integrators (Shaul et al., 2024a;b); S4S matches a teacher solver (Frankel et al., 2025); and a differentiable solver search jointly optimizes coefficients and time points (Wang et al., 2025c). Learning to Integrate Diffusion ODEs uses derivative-integral consistency (Liu & Yue, 2025), while ZEUS uses a second-order predictor to reduce denoiser evaluations (Wang et al., 2026). Geometry-based corrections use trajectory curvature and implicit denoising structure (Chen et al., 2024a), learned low-dimensional directions (Zhou et al., 2024), or PCA-based directions (Wang et al., 2025a). Our method represents acceleration in the state-velocity span using two time-dependent scalar coefficients without replacing the base field.

A separate line optimizes timestep placement using stochastic objectives (Sabour et al., 2024), reference trajectories (Xue et al., 2024), trajectory regularity (Chen et al., 2024a), convergence-based adaptive stepping (Chen et al., 2024b), and learned discretization (Tong et al., 2025). Other criteria include stiffness-derived error signals (Kong et al., 2026), instance-specific grids (Yuan et al., 2026), rotation-based schedules (Zhou et al., 2026), and joint solver-schedule design (Jo & Choi, 2026). SharpEuler converts finite-difference acceleration estimates into a fixed Euler grid (Gupta et al., 2026). Our method instead uses the structural residual as both the coefficient-fitting error and the scheduling signal; the resulting timestep schedule also applies to other integrators.

3 METHOD

3.1 PROBLEM SETTING: DATA-NOISE INTERPOLANTS

Let S be an optional conditioning variable and fix $S = s$, with $s = \emptyset$ denoting unconditional generation. Let $X_0 \sim p_{\mathrm{data}}(\cdot \mid s)$ and let $\xi \sim p_{\mathrm{ref}}$ be independent. We consider the forward interpolant, whose *state* X_t is given by $X_t = \alpha(t)X_0 + \sigma(t)\xi$, $t \in [0, T]$, and α and σ are smooth schedules satisfying $\alpha(0) = \sigma(T) = 1$ and $\alpha(T) = \sigma(0) = 0$. This formulation encompasses both the linear rectified-flow path and nonlinear time parameterizations (Liu et al., 2023). Taking derivatives with respect to time t, we obtain the *pathwise forward-interpolant velocity* as

$$V_t := \dot{X}_t = \dot{\alpha}(t)X_0 + \dot{\sigma}(t)\xi\,.$$

Let $p_t(\cdot \mid s)$ denote the marginal distribution of X_t. The population velocity field is the conditional mean defined by $v^*(z, t, s) := \mathbb{E}[V_t \mid X_t = z, S = s]$. The flow generated by

$$\dot{z}(t) = v^*(z(t), t, s) \tag{1}$$

transports the marginals $p_t(\cdot \mid s)$: backward integration maps $p_T = p_{\mathrm{ref}}$ to $p_0 = p_{\mathrm{data}}$.

Let v_θ be a pretrained approximation to v^*, and let $0 = t_0 < \cdots < t_N = T$ with $\Delta_n := t_{n+1} - t_n$ be a timestep grid. Writing y_n for the numerical approximation to $z_n = z(t_n)$, reverse Euler gives

$$y_n = y_{n+1} - \Delta_n v_\theta(y_{n+1}, t_{n+1}, s).$$

The acceleration along an exact reverse trajectory is the total derivative of the velocity:

$$a^*(z, t, s) := \partial_t v^*(z, t, s) + D_z v^*(z, t, s)v^*(z, t, s)\,, \tag{2}$$

where $D_z v^*$ is the Jacobian with respect to z. This acceleration determines the leading local truncation term of Euler and can be approximated without derivative evaluations during sampling.

3.2 FORWARD-INTERPOLANT MOTIVATION AND STRUCTURED ACCELERATION

Assume that $\alpha\dot{\sigma} - \dot{\alpha}\sigma \neq 0$. Because X_t and $\dot{X}_t$ depend on the same pair (X_0, ξ), eliminating this pair from $(X_t, \dot{X}_t, \ddot{X}_t)$ gives the *forward analytic backbone*:

$$\ddot{X}_t = A_{\mathrm{bb}}(t)X_t + B_{\mathrm{bb}}(t)\dot{X}_t, \tag{3}$$

where the subscript "bb" denotes the analytic backbone coefficients

$$A_{\mathrm{bb}}(t) = (\ddot{\alpha}\dot{\sigma} - \ddot{\sigma}\dot{\alpha})/(\alpha\dot{\sigma} - \dot{\alpha}\sigma), \quad B_{\mathrm{bb}}(t) = (\ddot{\sigma}\alpha - \ddot{\alpha}\sigma)/(\alpha\dot{\sigma} - \dot{\alpha}\sigma). \tag{4}$$

Appendix A.1 gives the derivation. Equation (3) indicates that *the forward acceleration lies in the span of the state and velocity* and motivates representing reverse acceleration in the same span. Applying the analytic coefficients to the learned field gives

$$a_{\mathrm{bb},\theta}(z, t, s) := A_{\mathrm{bb}}(t)z + B_{\mathrm{bb}}(t)v_\theta(z, t, s). \tag{5}$$

Using (5) as the second-order Taylor correction, we obtain

$$y_n = y_{n+1} - \Delta_n v_\theta(y_{n+1}, t_{n+1}, s) + \tfrac{\Delta_n^2}{2} a_{\mathrm{bb},\theta}(y_{n+1}, t_{n+1}, s). \tag{6}$$

The VP-cosine parameterization provides a revealing case. With $\alpha(t) = \cos t$, $\sigma(t) = \sin t$, and $t \in [0, \frac{\pi}{2}]$, the forward backbone is $\ddot{X}_t = -X_t$, so $(A_{\mathrm{bb}}, B_{\mathrm{bb}}) = (-1, 0)$, and (6) becomes

$$y_n = y_{n+1} - \Delta_n v_\theta(y_{n+1}, t_{n+1}, s) - \tfrac{\Delta_n^2}{2} y_{n+1}. \tag{7}$$

Expressed in the v-parameterization, deterministic DDIM agrees with (7) through second order (Appendix A.2). Thus, the forward-derived structure recovers the local DDIM update.

Because $A_{\mathrm{bb}}(t)$ and $B_{\mathrm{bb}}(t)$ need not describe the acceleration induced by the learned reverse velocity field v_θ, we introduce a coefficient head h_ϕ, parameterized by ϕ, that predicts two scalar coefficients: $(A_\phi(t, s), B_\phi(t, s)) := h_\phi(t, s)$. The resulting structured approximation to the acceleration of the learned field is

$$a_{\phi,\theta}(z, t, s) := A_\phi(t, s)z + B_\phi(t, s)v_\theta(z, t, s)\,,$$

which, when substituted into (6), gives the *structured-acceleration sampler*:

$$y_n = y_{n+1} - \Delta_n v_\theta(y_{n+1}, t_{n+1}, s) + \tfrac{\Delta_n^2}{2} a_{\phi,\theta}(y_{n+1}, t_{n+1}, s). \tag{8}$$

The sampler (8) requires only one velocity-field evaluation per step. The coefficient head adds only a multilayer perceptron (MLP) evaluation, which is negligible relative to the velocity-field evaluation. Setting $(A_\phi, B_\phi) = (0, 0)$ recovers Euler, while $(A_\phi, B_\phi) = (A_{\mathrm{bb}}, B_{\mathrm{bb}})$ recovers (6).

3.3 STRUCTURAL RESIDUAL AND GLOBAL ERROR

We define the *structural residual* as the difference between the reverse and structured accelerations:

$$R^*_{\phi,\theta}(z,t,s) := a^*(z,t,s) - a_{\phi,\theta}(z,t,s). \tag{9}$$

Proposition 1 (Local one-step defect)**.** *Fix s, let $z(t)$ solve* (1)*, and initialize one step of* (8) *with $y_{n+1} = z_{n+1}$. Define $\delta v_{n+1} := v_\theta(z_{n+1}, t_{n+1}, s) - v^*(z_{n+1}, t_{n+1}, s)$. Then*

$$z_n - y_n = \Delta_n \delta v_{n+1} + \tfrac{\Delta_n^2}{2} R^*_{\phi,\theta}(z_{n+1}, t_{n+1}, s) + O(\Delta_n^3). \tag{10}$$

The velocity-model and structural-residual terms scale as Δ_n and Δ_n^2, respectively. Appendix A.3 gives the proof.

Because a^* is unknown, $R^*_{\phi,\theta}$ cannot be computed directly. Instead, we define the acceleration of the learned field as

$$a_\theta(z,t,s) := \partial_t v_\theta(z,t,s) + D_z v_\theta(z,t,s) v_\theta(z,t,s),$$

and the *computable structural residual* as

$$R^\theta_{\phi,\theta}(z,t,s) := a_\theta(z,t,s) - a_{\phi,\theta}(z,t,s).$$

Fix s and suppress it from the notation below. Assume that v_θ is uniformly L-Lipschitz in its state argument, a^* is Lipschitz along reverse trajectories, and $\|A_\phi\|_\infty \le M_A$, $\|B_\phi\|_\infty \le M_B$. Define

$$\gamma_n := 1 + L\Delta_n, \qquad \Lambda_k := \prod_{j=0}^{k-1} \gamma_j,\ \Lambda_0 := 1, \qquad \Delta_{\max} := \max_n \Delta_n.$$

Theorem 2 (Global error bound)**.** *Draw $Z_{t_N} \sim p_{\mathrm{ref}}$, and let $\{Z_{t_n}\}_{n=0}^N$ be the exact reverse trajectory. Generate $\{y_n\}_{n=0}^N$ using* (8) *with $y_N = Z_{t_N}$. Let $\mathbb{P}_{\mathrm{rev}}$ be the joint distribution of these trajectories, and set $e_n := y_n - Z_{t_n}$. Define*

$$\mathcal{E}_n^{\mathrm{net}} := \|(v_\theta - v^*)(Z_{t_{n+1}}, t_{n+1})\|, \qquad \mathcal{E}_n^{\mathrm{acc}} := \|(a_\theta - a^*)(Z_{t_{n+1}}, t_{n+1})\|.$$

Write $\mathbb{E}_k := \mathbb{E}_{p_{t_{k+1}}(\cdot|s)}$, let $\mathcal{T} := \{t_n\}_{n=0}^N$, and take all maxima below over $0 \le k \le N-1$. Define

$$\mathcal{B}_\theta(\mathcal{T}) := \max_k \mathbb{E}_k[\mathcal{E}_k^{\mathrm{net}}] + \Delta_{\max} \max_k \mathbb{E}_k[\mathcal{E}_k^{\mathrm{acc}}] + \Delta_{\max}^2,$$
$$\mathcal{J}_\theta(\phi,\mathcal{T}) := \max_k \Lambda_k \Delta_k^2 \mathbb{E}_k \big\| R^\theta_{\phi,\theta}(Z_{t_{k+1}}, t_{k+1}) \big\|.$$

Then there exist constants $C_T, C_{T,N} > 0$, independent of ϕ, the timestep locations, and the residual values, such that

$$\mathbb{E}_{\mathbb{P}_{\mathrm{rev}}} \|e_0\| \le C_T \mathcal{B}_\theta(\mathcal{T}) + C_{T,N} \mathcal{J}_\theta(\phi,\mathcal{T}). \tag{11}$$

Here each $\mathbb{E}_k$ is taken over the exact marginal $p_{t_{k+1}}(\cdot \mid s)$, rather than over numerical sampler states. For fixed v_θ, the head affects the bound only through $\mathcal{J}_\theta$; $\mathcal{B}_\theta$ contains model mismatch and Taylor error. We therefore use $\mathcal{J}_\theta$ for coefficient fitting and timestep allocation. Appendix A.4 gives the proof.

3.4 RESIDUAL FITTING AND SCHEDULING FOR GUIDED SAMPLING

For fixed θ, $\mathcal{J}_\theta(\phi,\mathcal{T})$ depends on the head parameters ϕ and the timestep grid $\mathcal{T}$. We fit ϕ to reduce the residual magnitude, then select $\mathcal{T}$ to reduce the largest propagated residual contribution.

Define the unconditional and conditional fields by

$$v_\theta^{\mathrm{u}}(z,t) := v_\theta(z,t,\emptyset), \qquad v_\theta^{\mathrm{c}}(z,t,s) := v_\theta(z,t,s),$$

and, for a guidance scale w, define

$$v_{\theta,w}(z,t,s) := v_\theta^{\mathrm{u}}(z,t) + w\,(v_\theta^{\mathrm{c}}(z,t,s) - v_\theta^{\mathrm{u}}(z,t))\,. \tag{12}$$

The cases $w = 0$ and $w = 1$ correspond to unconditional and conditional sampling; other values give classifier-free guidance (CFG). For fixed (s,w), define

$$a_{\theta,w}(z,t,s) := \partial_t v_{\theta,w}(z,t,s) + D_z v_{\theta,w}(z,t,s) v_{\theta,w}(z,t,s). \tag{13}$$

Using guidance-aware coefficients $A_\phi(t,s,w)$ and $B_\phi(t,s,w)$, define

$$a_{\phi,\theta,w}(z,t,s) := A_\phi(t,s,w)z + B_\phi(t,s,w)v_{\theta,w}(z,t,s),$$
$$R^\theta_{\phi,\theta,w}(z,t,s) := a_{\theta,w}(z,t,s) - a_{\phi,\theta,w}(z,t,s).$$

Appendix A.5 discusses the head design. With θ fixed, we fit the coefficient head by minimizing

$$\mathcal{L}(\phi) := \mathbb{E}_{t,S,W,X_t}\left[\left\|R^\theta_{\phi,\theta,W}(X_t,t,S)\right\|\right], \qquad X_t \sim p_t(\cdot\,|\,S). \tag{14}$$

The variable W may be fixed at one guidance scale or distributed over several scales. Appendix A.6 derives the guided acceleration targets, and Algorithm 1 gives the offline pipeline. For CFG, the structured-acceleration correction reuses the conditional and unconditional evaluations in (12) and requires no additional velocity-field evaluations. For fixed (s,w), applying Theorem 2 to $v_{\theta,w}$, with expectations taken over the corresponding guided-flow marginals, gives

$$\mathcal{J}_{\theta,w}(\phi,\mathcal{T}) := \max_k \Lambda_k \Delta_k^2 \mathbb{E}_k \left\|R^\theta_{\phi,\theta,w}(Z_{t_{k+1}},t_{k+1})\right\|.$$

For $w \in \{0,1\}$, forward-interpolant samples from the corresponding marginals estimate these expectations without reverse trajectories. For other values of w, the guided-flow marginals are not directly available, so we use forward-interpolant samples as a proxy for these marginals. Given a fixed w and paired samples $\{(S^{(m)}, X_t^{(m)})\}_{m=1}^M$, define the *empirical residual profile*

$$\widehat{R}_w(t) := \tfrac{1}{M}\sum\nolimits_{m=1}^{M}\left\|R^\theta_{\phi,\theta,w}(X_t^{(m)},t,S^{(m)})\right\|. \tag{15}$$

A shared schedule may instead average the profile over W. On a candidate grid $0=\tau_0<\cdots<\tau_K=T$, define the interval-averaged residual

$$Q^{(w)}_{k,\ell} := \tfrac{1}{\tau_\ell-\tau_k}\sum\nolimits_{r=k+1}^{\ell}(\tau_r-\tau_{r-1})\widehat{R}_w(\tau_r), \qquad k<\ell.$$

Although Theorem 2 uses a pointwise residual, $Q^{(w)}_{k,\ell}$ averages the empirical profile over each candidate interval $[\tau_k,\tau_\ell]$ to reduce estimation noise. Define

$$c_w(k,\ell) := (\tau_\ell-\tau_k)^2 Q^{(w)}_{k,\ell}, \qquad \Gamma(k,\ell) := 1 + L(\tau_\ell-\tau_k)\,,$$

where L is a (spatial) Lipschitz constant. In the scheduling objective, $L \geq 0$ is a propagation parameter selected as described in Section 4. The cost c_w combines approximations of the residual and squared-step factors in $\mathcal{J}_{\theta,w}$, while products of Γ approximate its propagation factors.

For N sampling steps, the *residual-aware schedule* selects $0=k_0<k_1<\cdots<k_N=K$ by solving

$$\min_{0=k_0<\cdots<k_N=K}\max_{0\leq i\leq N-1}\left\{\left[\prod\nolimits_{j=0}^{i-1}\Gamma(k_j,k_{j+1})\right]c_w(k_i,k_{i+1})\right\}, \tag{16}$$

where the empty product is one. Appendix A.7 and Algorithm 2 specify the dynamic program and backtracking procedure.

3.5 Sampler and schedule variants

The structured-acceleration correction and the residual-aware schedule are two distinct components that may be used together or separately. Combining (8) with the schedule constructed from its matched residual $R^\theta_{\phi,\theta,w}$ yields a *structured-acceleration sampler with a residual-aware schedule*. In this case, the coefficient head serves two roles: it defines the acceleration correction and, through its matched residual, guides timestep allocation.

The same structured acceleration can also be incorporated into Heun's method as a second-order correction to its Euler predictor. For the guided velocity field $v_{\theta,w}$, we form the structured predictor

$$\widetilde{y}_n = y_{n+1} - \Delta_n v_{\theta,w}(y_{n+1},t_{n+1},s) + \tfrac{\Delta_n^2}{2}a_{\phi,\theta,w}(y_{n+1},t_{n+1},s), \tag{17}$$

followed by the Heun correction

$$y_n = y_{n+1} - \tfrac{\Delta_n}{2}\left[v_{\theta,w}(y_{n+1},t_{n+1},s) + v_{\theta,w}(\widetilde{y}_n,t_n,s)\right]. \tag{18}$$

Proposition 3 in Appendix A.8 shows that this method retains Heun's second-order accuracy under the stated regularity conditions.

For the structured predictor, $R^{\theta}_{\phi,\theta,w}$ defines the matched residual schedule. For methods without the head correction, set $A_\phi = B_\phi = 0$, so $a_{\phi,\theta,w} = 0$. The residual then reduces to $R^{\theta}_{\mathrm{Euler},\theta,w}(z,t,s) := a_{\theta,w}(z,t,s)$, which represents the acceleration term omitted by Euler. Replacing $R^{\theta}_{\phi,\theta,w}$ with $R^{\theta}_{\mathrm{Euler},\theta,w}$ in (15) gives an empirical Euler-residual profile, and the same minimax problem defines the Euler-residual schedule. This schedule applies to ordinary Heun, the second-order Adams–Bashforth method (AB2), and RF-Solver (Wang et al., 2025b). Thus, the learned-head residual is matched to the structured correction; the Euler residual depends only on the learned velocity field and can be transferred to other integrators. For structured-predictor Heun, we test schedules based on either residual because the structured correction modifies only the predictor.

3.6 Connections and scope

GENIE distills a data-dimensional JVP term, whereas RF-Solver constructs higher-order Taylor updates (Dockhorn et al., 2022; Wang et al., 2025b). Bespoke solvers and differentiable solver-search methods optimize model-specific solver coefficients and, in some cases, timesteps using trajectory-level objectives (Shaul et al., 2024a;b; Wang et al., 2025c). Our method freezes the base field and learns two scalar coefficients in the state-velocity span. The remaining acceleration component defines the structural residual used for error analysis and schedule construction. This construction requires no additional base-model evaluations during sampling.

AYS, GITS, and SharpEuler optimize fixed global grids using, respectively, a stochastic objective, trajectory geometry, and an offline acceleration profile (Sabour et al., 2024; Chen et al., 2024a; Gupta et al., 2026). Our schedule uses the structural residual from coefficient fitting. Its objective weights each interval cost by a product of propagation factors Γ, following the form in Theorem 2. Prior work traces signed one-step residuals through linearized dynamics to analyze spatial transport of endpoint error (Yin, 2026); our factors weight temporal contributions when selecting a fixed grid.

Our method does not select timesteps adaptively for individual samples. The global error theorem applies to the structured one-evaluation update. Applying the Euler-residual schedule to Heun, AB2, RF-Solver, or other integrators changes only the timestep grid and does not establish convergence results for those integrators. Further scope qualifications, limitations, and details on computational complexity are provided in Appendix A.10.

4 Experiments

We evaluate structured acceleration and timestep scheduling on image-space CIFAR-10 using a U-Net and on latent-space ImageNet-256 using SiT (Ma et al., 2024), covering unconditional, conditional, and generation with CFG. The baselines are Euler, Heun, AB2, and RF-Solver (Wang et al., 2025b). For the DSS comparison (Wang et al., 2025c), we match the SiT checkpoint, guidance scale $w = 1.375$, and evaluation budgets used in Table 1 of that work; we label the authors' DSS results as reported. To isolate acceleration and scheduling effects, we test structured acceleration on a uniform grid, Euler with an Euler-residual schedule, and their combination: structured acceleration with its matched residual-aware schedule. For Heun, we compare standard and structured predictors on uniform and residual-aware grids. For AB2 and RF-Solver, we transfer the Euler-residual schedule without changing their second-order updates. We treat L as a hyperparameter and use a grid sweep with small-sample FID to guide its selection. We compute the metrics using 50,000 samples.

CIFAR-10 VP-cosine special case. We test the VP-cosine case highlighted in Section 3.2, where $(A_{\mathrm{bb}}, B_{\mathrm{bb}}) = (-1, 0)$; network and checkpoint details are provided in Appendix A.9.1. The Euler residual is a_θ, while the transferred analytic backbone gives the residual $a_\theta + X_t$. Using paired forward-interpolant samples, we compute $a_\theta = \partial_t v_\theta + D_z v_\theta v_\theta$ via a JVP and evaluate both residual norms at each time. Figure 1 shows that the backbone generally reduces the expected residual at interior times, although the gain vanishes or reverses near the endpoints. Proposition 1 shows that the local residual defect is scaled by Δ_k^2, while Theorem 2 propagates these contributions through $\mathcal{J}_\theta$ with weights $\Lambda_k \Delta_k^2$. Thus, residual reduction is most consequential on coarse, low-NFE grids. Consistent with this prediction, the backbone lowers FID relative to Euler at

low NFE, while the difference diminishes as the grid is refined. The backbone's close agreement with DDIM is consistent with their updates being equivalent through second order. Euler has higher recall at low NFE, indicating a fidelity-coverage tradeoff rather than uniform dominance, but the recall gap largely closes at higher NFE. This special case validates the residual-based motivation. Our main results instead use fitted coefficients and their matched residual.

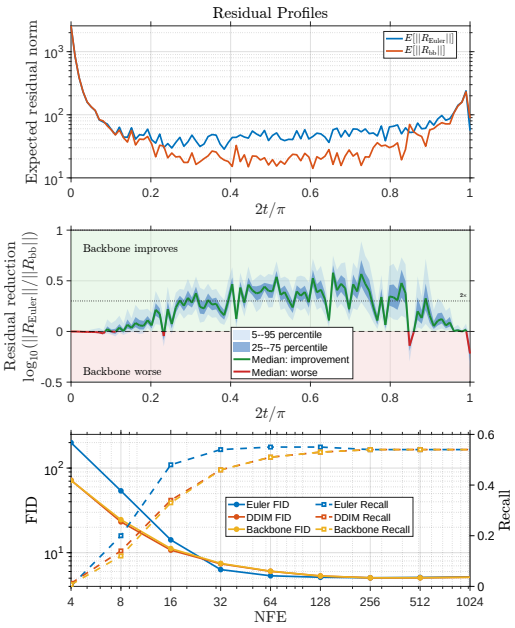

Figure 1: CIFAR-10 unconditional VP-cosine study. Top: residual profiles. Middle: paired residual reduction. Bottom: FID and recall versus NFE for Euler, DDIM, and the backbone. Residual statistics use $\Delta t = \pi/200$ and 5,000 paired samples per time.

CIFAR-10 rectified-flow models. We use three pretrained CIFAR-10 vector fields from Rectified Flow (Liu et al., 2023): 1-Rectified Flow (1-RF), 2-Rectified Flow (2-RF), and 3-Rectified Flow (3-RF). Starting from the baseline samplers, we evaluate the structured acceleration produced by the lightweight coefficient head, the dynamic-programming (DP) schedule constructed from the empirical residual profile, and their combination. These variants separate the effects of the structured sampler and timestep allocation. In plot legends and table entries, we use EUL, HEU, AB2, and RFS for the Euler, Heun, AB2, and RF-Solver (Wang et al., 2025b) baselines, and SAS and SPH for the structured-acceleration sampler and structured-predictor Heun. We use U, DPA, and DPR for the uniform, DP-acceleration, and DP-residual schedules, respectively. Specifically, DPA uses the empirical profile in (15) with $R^{\theta}_{\phi,\theta}$ replaced by a_θ, whereas DPR uses the learned-head residual profile as defined in (15). Appendix A.9.2 provides checkpoint and implementation details, including the architecture of the lightweight coefficient head and the selection of L, as well as residual profiles, paired residual-reduction plots, and fitted coefficient profiles. Figure 4 in Appendix A.9.2 compares the DPA and DPR sampling schedules with the uniform schedule.

Table 1 separates the contributions of the numerical update and timestep allocation through comparisons within each baseline-solver block at matched NFE, yielding the following observations.

Euler block. Structured acceleration alone (SAS-U) lowers FID and increases recall versus EUL-U at every budget. Scheduling alone (EUL-DPA) lowers FID at every budget, with slightly lower recall at NFE 4–256 and a tie at NFE 512. Together, SAS-DPR achieves the lowest FID in the block at NFE 4 and 32–512, with ties included. Matching the schedule to the learned residual lowers FID relative to SAS-DPA at NFE 4–128, with ties thereafter. However, SAS-U has the lowest FID at NFE 8 and 16 and the highest recall throughout the block, illustrating a fidelity-coverage tradeoff.

Heun block. Structured prediction alone (SPH-U) modestly lowers FID versus HEU-U at NFE 4–128. Transferring DPA without changing Heun's update yields larger gains. Combining structured prediction with DPA gives further gains; recall improves upon or ties HEU-DPA throughout. For SPH, DPA gives lower FID than DPR at NFE 4–32, whereas DPR is slightly better at NFE 64–512.

AB2 and RF-Solver blocks. Transferring DPA improves sampling without modifying either solver's update, lowering AB2 FID at NFE 4–256 and RF-Solver FID at NFE 8–128. Recall remains close to the uniform baselines, with absolute differences of at most 0.9 percentage points for AB2 and 3.3 for RF-Solver. RF-Solver's gains are concentrated at intermediate budgets; at NFE 256 and 512, FID differs from the uniform baseline by only 0.01 and 0.02, respectively.

Overall, these results demonstrate the value of structured acceleration and structured prediction in improving their respective baselines, as well as the value of DP scheduling in improving FID across multiple solver families. The gains are most pronounced at low-to-intermediate NFE. At high NFE, performance differences narrow. Results for 2-RF and 3-RF are provided in Table 4 in Appendix A.9.2.

Table 1: CIFAR-10 unconditional sampling with the 1-RF model. Methods are grouped by baseline sampler, with uniform-schedule baseline names in bold; columns report NFE. Bold values indicate the best result within each solver block at each NFE (lowest FID or highest recall), including ties at the displayed precision. Gray-shaded FID cells mark the lowest tested NFE at which each variant matches or improves on its solver block's uniform baseline at NFE 512. DPA and DPR use $L = 2.0$ at NFE 4 and $L = 1.0$ at all other budgets; see Appendix A.9.2 for the selection procedure.

	Sampler-Schedule	4	8	16	32	64	128	256	512
(a) FID ↓	**EUL-U**	52.0	16.6	8.11	5.09	3.73	3.12	2.87	2.76
	EUL-DPA	49.8	16.2	7.02	4.23	3.22	2.88	2.77	2.73
	SAS-U	50.5	**7.88**	**4.68**	3.79	3.21	2.91	2.76	2.71
	SAS-DPA	37.2	10.1	5.14	3.51	2.99	2.80	**2.72**	**2.70**
	SAS-DPR	**28.1**	9.07	5.07	**3.50**	**2.97**	**2.79**	**2.72**	**2.70**
	HEU-U	302.	129.	57.3	23.1	7.10	3.05	**2.66**	2.67
	HEU-DPA	301.	77.1	12.3	3.09	2.68	2.67	2.68	2.68
	SPH-U	**286.**	111.	55.4	22.4	6.96	3.04	**2.66**	2.67
	SPH-DPA	291.	**58.4**	**9.54**	**2.95**	2.67	2.67	2.68	2.68
	SPH-DPR	292.	74.3	13.0	3.26	**2.66**	**2.66**	**2.66**	**2.66**
	AB2-U	29.1	8.94	5.58	3.93	3.10	2.82	2.72	**2.68**
	AB2-DPA	**26.9**	**8.50**	**4.46**	**3.13**	**2.77**	**2.70**	**2.68**	**2.68**
	RFS-U	**35.7**	12.0	6.78	4.45	3.13	2.72	**2.65**	**2.64**
	RFS-DPA	36.5	**8.97**	**4.75**	**3.02**	**2.69**	**2.66**	2.66	2.66
(b) Recall (%)↑	**EUL-U**	2.08	17.7	37.7	49.1	53.4	55.4	56.3	56.6
	EUL-DPA	1.62	16.6	35.9	47.6	52.6	55.1	55.9	56.6
	SAS-U	**20.7**	**36.6**	**46.5**	**52.0**	**54.7**	**56.0**	**56.6**	**56.9**
	SAS-DPA	9.87	26.8	41.1	49.8	53.6	55.5	56.1	56.7
	SAS-DPR	9.37	30.9	42.3	50.1	53.8	55.5	56.2	56.7
	HEU-U	0.00	0.05	8.16	38.4	53.7	56.6	56.9	57.0
	HEU-DPA	0.00	3.56	46.4	55.5	**56.8**	56.9	57.0	57.0
	SPH-U	0.00	0.09	8.41	38.9	53.6	56.6	56.9	57.1
	SPH-DPA	0.00	**9.59**	**48.7**	55.8	**56.8**	56.9	57.0	57.1
	SPH-DPR	0.00	3.37	46.7	**55.9**	56.6	**57.0**	**57.1**	**57.2**
	AB2-U	11.2	41.4	**53.8**	**57.1**	**57.3**	**57.2**	**57.1**	**57.2**
	AB2-DPA	**12.1**	**41.5**	53.0	56.3	56.9	57.0	57.0	57.0
	RFS-U	**16.7**	**40.9**	**52.8**	**56.9**	**57.2**	**57.1**	**57.0**	57.0
	RFS-DPA	15.6	37.6	51.5	55.8	56.7	56.9	56.9	**57.1**

Abbreviations.
EUL: Euler
HEU: Heun
AB2: second-order Adams–Bashforth
RFS: RF-Solver
SAS: structured-acceleration sampler
SPH: structured-predictor Heun
U: uniform
DPA: DP-acceleration
DPR: DP-residual

Table 2: ImageNet-256 sampling with SiT-XL at $w = 1$ (no CFG) and $w = 1.5$. NFE counts both CFG branches; bold names denote uniform baselines. SiT's reported Heun ODE reference results have FID/Recall 9.35/68% at NFE 250 and 2.15/60% at NFE 500, respectively (Ma et al., 2024); the NFE 500 result applies CFG only to the first three of the four latent output channels. Bold values indicate the best result within each solver block at each NFE (lowest FID or highest recall), including ties at the displayed precision. N/E: not evaluated.

	Sampler-	$w = 1$			$w = 1.5$		
	Schedule	16	64	250	32	128	500
(a) FID ↓	**EUL-U**	16.9	**10.5**	**9.66**	3.24	**2.15**	**2.11**
	SAS-DPR	**15.9**	10.7	9.71	**3.06**	2.19	2.12
	HEU-U	15.4	9.76	**9.42**	2.97	**2.10**	**2.11**
	SPH-DPA	**14.3**	9.66	N/E	**2.73**	2.12	N/E
	SPH-DPR	14.6	**9.65**	**9.42**	2.97	2.11	**2.11**
	AB2-U	11.4	9.54	N/E	2.24	**2.10**	N/E
	AB2-DPA	**9.46**	**9.47**	N/E	**2.15**	**2.10**	N/E
	RFS-U	11.4	9.60	N/E	2.33	**2.10**	N/E
	RFS-DPA	**9.72**	**9.49**	N/E	**2.25**	2.11	N/E
(b) Recall (%) ↑	**EUL-U**	64.2	**67.0**	66.9	54.1	57.4	58.1
	SAS-DPR	**65.3**	**67.0**	**67.1**	**56.0**	**57.6**	**58.3**
	HEU-U	65.6	67.1	**67.3**	55.3	**58.2**	**58.4**
	SPH-DPA	**65.9**	**67.3**	N/E	**56.4**	**58.2**	N/E
	SPH-DPR	65.7	67.2	67.2	56.3	**58.2**	58.0
	AB2-U	66.8	67.1	N/E	56.6	**58.2**	N/E
	AB2-DPA	**66.9**	**67.4**	N/E	**57.3**	57.8	N/E
	RFS-U	**67.7**	67.2	N/E	**57.4**	**58.4**	N/E
	RFS-DPA	67.2	**67.3**	N/E	57.2	58.1	N/E

ImageNet-256 latent-space experiments. We evaluate pretrained SiT-XL (Ma et al., 2024) at $w \in \{1, 1.375, 1.5\}$, with no CFG at $w = 1$. For CFG, NFE counts both branches. Table 2 compares nine variants based on Euler, Heun, AB2, and RF-Solver at NFE 16 and 64 for $w = 1$ and NFE 32 and 128 for $w = 1.5$. At NFE 250 and 500, evaluations are restricted to SAS-DPR/EUL-U and SPH-DPR/HEU-U; we rerun HEU-U using the pipeline of Ma et al. (2024). Table 3 compares six feasible variants against DSS at $w = 1.375$ and matched NFE $\{12, 14, 16, 20, 30\}$. Appendix A.9.3 gives residual profiles and implementation details, Appendix A.10 reports offline costs, and Appendix A.9.4 studies sensitivity to L.

Table 2 shows a low-budget advantage across all four solver families. At the smallest budget for each guidance setting, the residual-guided variant lowers FID for Euler, Heun, AB2, and RF-Solver. At $w = 1$, FID improves from 16.9 to 15.9 for Euler, 15.4 to 14.3 for Heun, 11.4 to 9.46 for AB2, and

Table 3: ImageNet-256 comparison with DSS at $w = 1.375$. Columns report total NFE, counting both CFG branches. DSS values are reported in Table 1 of Wang et al. (2025c). Bold values indicate the best result within each solver block at each NFE. Gray-shaded cells indicate sampler-schedule combinations that outperform DSS in FID or IS. †: the sampler-schedule combination cannot run at the given NFE.

	Sampler-Schedule	12	14	16	20	30
(a) FID ↓	**HEU-U**	**38.6**	†	15.5	8.67	†
	SPH-DPA	39.7	†	**15.4**	**8.28**	†
	AB2-U	6.39	5.06	4.24	3.40	2.57
	AB2-DPA	**4.03**	**3.21**	**2.81**	**2.44**	**2.17**
	RFS-U	19.2	†	7.58	4.37	†
	RFS-DPA	**9.66**	†	**4.11**	**2.97**	†
	DSS (reported)	3.57	2.78	2.65	2.40	2.24
(b) IS ↑	**HEU-U**	**77.8**	†	**149.**	**187.**	†
	SPH-DPA	73.0	†	145.	**187.**	†
	AB2-U	198.	212.	221.	232.	246.
	AB2-DPA	**217.**	**230.**	**238.**	**248.**	**257.**
	RFS-U	97.4	†	164.	202.	†
	RFS-DPA	**144.**	†	**208.**	**233.**	†
	DSS (reported)	214.	229.	234.	238.	244.

11.4 to 9.72 for RF-Solver. At $w = 1.5$, the corresponding improvements are 3.24 to 3.06, 2.97 to 2.73, 2.24 to 2.15, and 2.33 to 2.25. Thus, the benefit is not confined to the structured one-evaluation update: the acceleration-derived schedule also transfers effectively to existing second-order solvers without modifying their numerical updates. The recall results further distinguish the solver families. SAS-DPR improves or ties recall at every reported budget, while SPH-DPA improves or ties recall at all of its evaluated settings. AB2 also gains recall at the lowest budget under both guidance settings. RF-Solver exhibits small recall reductions despite improved low-budget FID, indicating a modest fidelity-coverage tradeoff. The FID gains generally narrow and occasionally reverse as NFE increases. This budget dependence is consistent with the leading residual contribution being weighted by the squared step size: timestep allocation is most consequential on coarse grids, whereas the methods converge to similar performance as discretization error decreases.

Table 3 shows that the benefit of DPA depends strongly on the underlying solver. For AB2, DPA improves both FID and IS at every tested NFE. Relative to AB2-U, AB2-DPA lowers FID from 6.39 to 4.03 at NFE 12 and from 2.57 to 2.17 at NFE 30, while increasing IS from 198 to 217 and from 246 to 257, respectively. AB2-DPA also achieves higher IS than DSS at every matched budget. Although its FID remains slightly worse than DSS at NFE 12–20, the gap decreases as NFE increases, and at NFE 30 AB2-DPA outperforms DSS in both FID (2.17 versus 2.24) and IS (257 versus 244). DPA also substantially improves RF-Solver at every feasible budget; for example, at NFE 20 it reduces FID from 4.37 to 2.97 and raises IS from 202 to 233, although DSS still performs better on both metrics. In contrast, the gains for Heun are limited: SPH-DPA slightly improves FID at NFE 16 and 20, but yields lower IS at NFE 12 and 16 and matches HEU-U at NFE 20. This solver dependence partly reflects how a fixed NFE budget translates into integration steps. Heun requires two vector-field evaluations per integration step per branch, whereas AB2 requires only one. Thus, with CFG, Heun uses four evaluations per step and AB2 uses two, so Heun takes roughly half as many integration steps at the same total NFE. At NFE 12, for example, Heun takes only three integration steps, resulting in a particularly coarse grid and larger discretization error. This helps explain why AB2 outperforms Heun at low NFE in both Tables 2 and 3. This is notable because DSS jointly optimizes model-specific solver coefficients and timesteps using a trajectory-level objective, whereas DPA changes only the timestep schedule of AB2 and RF-Solver, leaving the pretrained vector field and solver update fixed.

5 CONCLUSIONS

We introduced a residual-guided framework that unifies acceleration correction, error analysis, and timestep allocation for diffusion and flow sampling. Motivated by the pathwise forward-acceleration identity for nondegenerate data-noise interpolants, we use the state-velocity span as an inductive bias and learn two scalar coefficients while keeping the velocity field fixed. The resulting structural residual guides both coefficient fitting and propagation-aware minimax scheduling. It characterizes the leading local defect relative to the learned field and enters a global error decomposition that separates sampler-dependent error from model mismatch. The structured-acceleration correction requires no additional velocity-field evaluations during sampling, and the schedules transfer to existing integrators. Empirical gains vary across solvers and budgets, with the clearest improvements at low-to-intermediate evaluation budgets and smaller differences on finer grids. Appendix A.10 discusses the method's limitations, the scope of the theoretical guarantees, and computational complexity.

AI USE STATEMENT

In this work, we used generative AI tools to assist with framing the work, refining research hypotheses, designing experiments, implementing code snippets for the experiments. We did not use generative AI tools to formulate mathematical claims and develop or prove mathematical arguments, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to assist with drafting, editing, and polishing the writing, including improving the clarity and presentation of proofs. The AI tools were used only to improve exposition and did not generate the underlying mathematical claims, proof strategies, or substantive mathematical reasoning. We have reviewed all AI-assisted work. All AI-assisted text was carefully checked and revised by the authors for accuracy, clarity, and consistency with the intended technical content. In particular, AI-assisted edits to proofs were verified against the authors' original mathematical arguments to ensure that they did not alter the meaning or validity of the proofs. AI-assisted code snippets were reviewed, integrated, and tested by the authors to verify their correctness and consistency with the experimental methodology and reported results.

We take responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We are not aware of any specific ethical concerns raised by this work beyond those generally associated with research in machine learning.

REPRODUCIBILITY STATEMENT

We have taken steps to facilitate the reproducibility of our results. The paper and appendix provide detailed descriptions of the proposed methods, experimental settings, datasets, evaluation procedures, and implementation details. We provide an anonymized codebase as part of the supplementary materials, including the implementations and scripts needed to reproduce the experiments and results reported in this paper. Additional details, including hyperparameter settings and complete proofs of the theoretical results, are provided in the appendix.

AUTHOR CONTRIBUTIONS

Author contributions are omitted for double-blind review and will be included in the camera-ready version.

ACKNOWLEDGMENTS

Acknowledgments are omitted for double-blind review and will be included in the camera-ready version.

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025. URL https://jmlr.org/papers/v26/23-1605.html.

Defang Chen, Zhenyu Zhou, Can Wang, Chunhua Shen, and Siwei Lyu. On the trajectory regularity of ODE-based diffusion sampling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7905–7934. PMLR, 2024a. URL https://proceedings.mlr.press/v235/chen24bm.html.

Yuzhu Chen, Fengxiang He, Shi Fu, Xinmei Tian, and Dacheng Tao. Adaptive time-stepping schedules for diffusion models. In *Proceedings of the 40th Conference on Uncertainty in Artificial*

Intelligence, volume 244 of *Proceedings of Machine Learning Research*, pp. 685–697. PMLR, 2024b. URL https://proceedings.mlr.press/v244/chen24c.html.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. GENIE: Higher-order denoising diffusion solvers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30150–30166, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c281c5a17ad2e55e1ac1ca825071f991-Abstract.html.

Eric Frankel, Sitan Chen, Jerry Li, Pang Wei Koh, Lillian J. Ratliff, and Sewoong Oh. S4S: Solving for a fast diffusion model solver. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 17542–17574. PMLR, 2025. URL https://proceedings.mlr.press/v267/frankel25a.html.

Xiaomeng Fu, Jia Li, Yiming Hu, Yong Wang, Xi Wang, Hayden Kwok-Hay So, and Xiangxiang Chu. Acceleration-aware sampling for few-step rectified flow models. OpenReview preprint, 2025. URL https://openreview.net/forum?id=hCIdPgTfUw.

Aditi Gupta, Soon Hoe Lim, Annan Yu, and N. Benjamin Erichson. Sharpen your flow: Sharpness-aware sampling for flow matching. *arXiv preprint arXiv:2605.11547*, 2026. doi: 10.48550/arXiv.2605.11547. URL https://arxiv.org/abs/2605.11547.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Sangwoo Jo and Sungjoon Choi. Formalizing the sampling design space of diffusion-based generative models via adaptive solvers and wasserstein-bounded timesteps. *arXiv preprint arXiv:2602.12624*, 2026. URL https://arxiv.org/abs/2602.12624.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

Inho Kong, Sojin Lee, Youngjoon Hong, and Hyunwoo J. Kim. Error as signal: Stiffness-aware diffusion sampling via embedded runge-kutta guidance. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wKX9OL0leb.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PlKWVd2yBkY.

Wenze Liu and Xiangyu Yue. Learning to integrate diffusion odes by averaging the derivatives. In *Advances in Neural Information Processing Systems*, volume 38, pp. 133825–133859, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c226d7bd4d9a3542a95c6a738e18463b-Paper-Conference.pdf.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

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. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5775–5787, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/260a14acce2a89dad36adc8eefe7c59e-Abstract.html.

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. doi: 10.1007/s11633-025-1562-4. URL https://arxiv.org/abs/2211.01095.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024. doi: 10.1007/978-3-031-72980-5_2. URL https://doi.org/10.48550/arXiv.2401.08740.

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.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42947–42975. PMLR, 2024. URL https://proceedings.mlr.press/v235/sabour24a.html.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2202.00512.

Neta Shaul, Juan Perez, Ricky T. Q. Chen, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke solvers for generative flow models. In *International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=1PXEY7ofFX.

Neta Shaul, Uriel Singer, Ricky T. Q. Chen, Matthew Le, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke non-stationary solvers for fast sampling of diffusion and flow models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 44603–44627. PMLR, 2024b. URL https://proceedings.mlr.press/v235/shaul24a.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Vinh Tong, Trung-Dung Hoang, Anji Liu, Guy Van den Broeck, and Mathias Niepert. Learning to discretize denoising diffusion ODEs. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/750046157471c56235a781f2eff6e226-Abstract-Conference.html.

Guangyi Wang, Wei Peng, Lijiang Li, Wenyu Chen, Yuren Cai, and Song-Zhi Su. Diffusion sampling correction via approximately 10 parameters. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64023–64043. PMLR, 2025a. URL https://proceedings.mlr.press/v267/wang25cd.html.

Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing. In *Proceedings of the 42nd*

International Conference on Machine Learning, volume 267 of *Proceedings of Machine Learning Research*, pp. 64044–64058. PMLR, 2025b. URL https://proceedings.mlr.press/v267/wang25ce.html.

Shuai Wang, Zexian Li, Qipeng Zhang, Tianhui Song, Xubin Li, Tiezheng Ge, Bo Zheng, and Limin Wang. Differentiable solver search for fast diffusion sampling. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63549–63568. PMLR, 2025c. URL https://proceedings.mlr.press/v267/wang25bh.html.

Yixiao Wang, Ting Jiang, Zishan Shao, Hancheng Ye, Jingwei Sun, Mingyuan Ma, Jianyi Zhang, Yiran Chen, and Hai Li. ZEUS: Accelerating diffusion models with only second-order predictor. *arXiv preprint arXiv:2604.01552*, 2026. doi: 10.48550/arXiv.2604.01552. URL https://arxiv.org/abs/2604.01552.

Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=VFBjuF8HEp.

Shuchen Xue, Zhaoqiang Liu, Fei Chen, Shifeng Zhang, Tianyang Hu, Enze Xie, and Zhenguo Li. Accelerating diffusion sampling with optimized time steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8292–8301, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Xue_Accelerating_Diffusion_Sampling_with_Optimized_Time_Steps_CVPR_2024_paper.html.

Songheng Yin. Spatial transport of integration error in generative odes. *arXiv preprint arXiv:2607.16361*, 2026. doi: 10.48550/arXiv.2607.16361. URL https://arxiv.org/abs/2607.16361.

Liangyu Yuan, Ruoyu Wang, Tong Zhao, Dingwen Fu, Mingkun Lei, Beier Zhu, and Chi Zhang. Few-step diffusion sampling through instance-aware discretizations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35882–35892, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Yuan_Few-Step_Diffusion_Sampling_Through_Instance-Aware_Discretizations_CVPR_2026_paper.html.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Loek7hfb46P.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49842–49869, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/9c2aa1e456ea543997f6927295196381-Abstract-Conference.html.

Wenliang Zhao, Minglei Shi, Xumin Yu, Jie Zhou, and Jiwen Lu. FlowTurbo: Towards real-time flow-based image generation with velocity refiner. In *Advances in Neural Information Processing Systems*, volume 37, pp. 4148–4176, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/07c256a163a7559186ec1c71e95b9ec9-Abstract-Conference.html.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. DPM-Solver-v3: Improved diffusion ODE solver with empirical model statistics. In *Advances in Neural Information Processing Systems*, volume 36, pp. 55502–55542, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ada8de994b46571bdcd7eeff2d3f9cff-Abstract-Conference.html.

Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ODE-based sampling for diffusion models in around 5 steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7777–7786, 2024. URL https://openaccess.

thecvf.com/content/CVPR2024/html/Zhou_Fast_ODE-based_Sampling_for_Diffusion_Models_in_Around_5_Steps_CVPR_2024_paper.html.

Zhenyu Zhou, Defang Chen, Siwei Lyu, Chun Chen, and Can Wang. Analyzing and improving fast sampling of text-to-image diffusion models. *arXiv preprint arXiv:2603.00763*, 2026. URL https://arxiv.org/abs/2603.00763.

A Appendix

A.1 Derivation of the forward acceleration identity

From the forward interpolant and its derivative,

$$\begin{pmatrix} X_t \\ \dot{X}_t \end{pmatrix} = \begin{pmatrix} \alpha & \sigma \\ \dot{\alpha} & \dot{\sigma} \end{pmatrix} \begin{pmatrix} X_0 \\ \xi \end{pmatrix},$$

where all schedule functions are evaluated at t. Let $D(t) := \alpha\dot{\sigma} - \dot{\alpha}\sigma$. Under the nondegeneracy condition $D(t) \neq 0$, inversion gives

$$X_0 = \frac{\dot{\sigma}X_t - \sigma\dot{X}_t}{D(t)}, \qquad \xi = \frac{-\dot{\alpha}X_t + \alpha\dot{X}_t}{D(t)}. \tag{19}$$

Differentiating the interpolant twice and substituting (19) yields

$$\begin{aligned} \ddot{X}_t &= \ddot{\alpha}X_0 + \ddot{\sigma}\xi \\ &= \frac{\ddot{\alpha}\dot{\sigma} - \ddot{\sigma}\dot{\alpha}}{\alpha\dot{\sigma} - \dot{\alpha}\sigma}X_t + \frac{\ddot{\sigma}\alpha - \ddot{\alpha}\sigma}{\alpha\dot{\sigma} - \dot{\alpha}\sigma}\dot{X}_t \\ &= A_{\mathrm{bb}}(t)X_t + B_{\mathrm{bb}}(t)\dot{X}_t, \end{aligned}$$

which proves (3) and (4).

A.2 Second-order equivalence to deterministic DDIM

Consider the cosine VP parameterization in angular time, $\alpha(t) = \cos t$ and $\sigma(t) = \sin t$. For any fixed endpoint pair (x_0, ξ), the corresponding interpolant and velocity (the tangent to the interpolant) are

$$x(t) = \cos(t)x_0 + \sin(t)\xi, \qquad \dot{x}(t) = -\sin(t)x_0 + \cos(t)\xi.$$

A v-prediction model estimates this velocity. Thus, at a reverse state y_t, the data and noise predictions $\widehat{x}_0$ and $\widehat{\xi}$ associated with $v_\theta(y_t, t, s)$ are defined by

$$\begin{aligned} y_t &= \cos(t)\widehat{x}_0 + \sin(t)\widehat{\xi}, \\ v_\theta(y_t, t, s) &= -\sin(t)\widehat{x}_0 + \cos(t)\widehat{\xi}. \end{aligned} \tag{20}$$

Equation (20) is the velocity parameterization: its first line reconstructs the current state, while its second identifies the model output with the velocity of the same predicted interpolant. Its coefficient matrix is a rotation, so inversion gives

$$\widehat{x}_0 = \cos(t)y_t - \sin(t)v_\theta(y_t, t, s), \qquad \widehat{\xi} = \sin(t)y_t + \cos(t)v_\theta(y_t, t, s).$$

For any earlier time $u < t$, deterministic DDIM (Song et al., 2021a) evaluates the predicted path at u using the endpoint predictions inferred from the current pair (y_t, t):

$$y_u^{\mathrm{DDIM}} = \alpha(u)\widehat{x}_0(y_t, t, s) + \sigma(u)\widehat{\xi}(y_t, t, s).$$

Setting $u = t - \Delta$ under the cosine VP parameterization gives

$$\begin{aligned} y_{t-\Delta}^{\mathrm{DDIM}} &= \cos(t-\Delta)\widehat{x}_0 + \sin(t-\Delta)\widehat{\xi} \\ &= \big[\cos(t-\Delta)\cos t + \sin(t-\Delta)\sin t\big]y_t \\ &\quad + \big[-\cos(t-\Delta)\sin t + \sin(t-\Delta)\cos t\big]v_\theta(y_t, t, s) \\ &= \cos(\Delta)y_t - \sin(\Delta)v_\theta(y_t, t, s), \end{aligned} \tag{21}$$

where the last equality uses $\cos(a-b) = \cos a \cos b + \sin a \sin b$ and $\sin((t-\Delta)-t) = -\sin\Delta$. Equation (21) is the angular DDIM update derived by Salimans & Ho (2022, Appendix D, Eq. (38)). Here we use its Taylor expansion to interpret the forward-derived acceleration correction within our framework. Specifically,

$$\cos\Delta = 1 - \frac{\Delta^2}{2} + O(\Delta^4), \qquad \sin\Delta = \Delta + O(\Delta^3).$$

Substitution into (21) yields

$$y_{t-\Delta}^{\mathrm{DDIM}} = y_t - \Delta v_\theta(y_t, t, s) - \frac{\Delta^2}{2} y_t + O(\Delta^3),$$

which is exactly (7) through second order.

A.3 Proof of the local one-step defect

Write $z_n := z(t_n)$. A backward Taylor expansion of the exact reverse trajectory gives

$$z_n = z_{n+1} - \Delta_n v^*(z_{n+1}, t_{n+1}, s) + \frac{\Delta_n^2}{2} a^*(z_{n+1}, t_{n+1}, s) + O(\Delta_n^3).$$

Initializing (8) at $y_{n+1} = z_{n+1}$ and subtracting the numerical update from this expansion yields

$$\begin{aligned} z_n - y_n = \Delta_n &[v_\theta(z_{n+1}, t_{n+1}, s) - v^*(z_{n+1}, t_{n+1}, s)] \\ &+ \frac{\Delta_n^2}{2} \left[a^*(z_{n+1}, t_{n+1}, s) - a_{\phi,\theta}(z_{n+1}, t_{n+1}, s)\right] + O(\Delta_n^3). \end{aligned}$$

Applying (9) proves (10).

A.4 Proof of the computable expected global error bound

Fix the condition s and suppress it from the notation. Recall that the admissible coefficient heads satisfy $\|A_\phi\|_\infty \le M_A$ and $\|B_\phi\|_\infty \le M_B$, uniformly in ϕ. We follow three steps: first derive a pathwise bound involving the exact structural residual, then take expectations, and finally replace the unknown exact residual by its computable counterpart.

Step 1: Pathwise bound with the exact residual. Let $z_n := z(t_n)$, $e_n := y_n - z_n$, and

$$\varepsilon_n^{\mathrm{net}} := \|v_\theta(z_{n+1}, t_{n+1}) - v^*(z_{n+1}, t_{n+1})\|.$$

With s suppressed, this definition gives $\varepsilon_n^{\mathrm{net}} = \|\delta v_{n+1}\|$. It is the pathwise realization of the random variable $\mathcal{E}_n^{\mathrm{net}}$ defined in Theorem 2. Set

$$K_{\mathrm{coef}} := \frac{1}{2}\left(M_A + L M_B\right), \qquad \Lambda_k := \prod_{j=0}^{k-1} \gamma_j, \qquad \Lambda_0 := 1.$$

For clarity, write the numerical update as

$$y_n = y_{n+1} - \Delta_n v_\theta(y_{n+1}, t_{n+1}) + \frac{\Delta_n^2}{2} a_{\phi,\theta}(y_{n+1}, t_{n+1}). \tag{22}$$

The exact reverse trajectory has the backward Taylor expansion

$$z_n = z_{n+1} - \Delta_n v^*(z_{n+1}, t_{n+1}) + \frac{\Delta_n^2}{2} a^*(z_{n+1}, t_{n+1}) + \rho_n, \tag{23}$$

where the time-Lipschitz assumption on the exact acceleration implies $\|\rho_n\| \le C\Delta_n^3$ for a constant C independent of n. By the definition of the exact structural residual,

$$a^*(z_{n+1}, t_{n+1}) = A_\phi(t_{n+1}) z_{n+1} + B_\phi(t_{n+1}) v_\theta(z_{n+1}, t_{n+1}) + R^*_{\phi,\theta}(z_{n+1}, t_{n+1}).$$

Equivalently, $a^* = a_{\phi,\theta} + R^*_{\phi,\theta}$. Substituting this decomposition into (23) gives

$$z_n = z_{n+1} - \Delta_n v^*(z_{n+1}, t_{n+1}) + \frac{\Delta_n^2}{2} a_{\phi,\theta}(z_{n+1}, t_{n+1}) + r_n. \tag{24}$$

This has the same structured-acceleration term as the numerical update (22); the remaining local defect is

$$r_n := \frac{\Delta_n^2}{2} R^*_{\phi,\theta}(z_{n+1}, t_{n+1}) + \rho_n.$$

Consequently,

$$\|r_n\| \le \frac{\Delta_n^2}{2} \|R^*_{\phi,\theta}(z_{n+1}, t_{n+1})\| + C\Delta_n^3, \tag{25}$$

We now compute the error directly from its definition. Substituting the numerical step (22) and the exact step (24) into $e_n = y_n - z_n$ gives

$$\begin{aligned}
e_n &= y_n - z_n \\
&= \left[y_{n+1} - \Delta_n v_\theta(y_{n+1}, t_{n+1}) + \frac{\Delta_n^2}{2} a_{\phi,\theta}(y_{n+1}, t_{n+1}) \right] \\
&\quad - \left[z_{n+1} - \Delta_n v^*(z_{n+1}, t_{n+1}) + \frac{\Delta_n^2}{2} a_{\phi,\theta}(z_{n+1}, t_{n+1}) + r_n \right] \\
&= (y_{n+1} - z_{n+1}) - \Delta_n \left[v_\theta(y_{n+1}, t_{n+1}) - v^*(z_{n+1}, t_{n+1}) \right] \\
&\quad + \frac{\Delta_n^2}{2} \left[a_{\phi,\theta}(y_{n+1}, t_{n+1}) - a_{\phi,\theta}(z_{n+1}, t_{n+1}) \right] - r_n \\
&= e_{n+1} - \Delta_n \left[v_\theta(y_{n+1}, t_{n+1}) - v^*(z_{n+1}, t_{n+1}) \right] \\
&\quad + \frac{\Delta_n^2}{2} \left[a_{\phi,\theta}(y_{n+1}, t_{n+1}) - a_{\phi,\theta}(z_{n+1}, t_{n+1}) \right] - r_n.
\end{aligned}$$

Next, the definition of the structured acceleration gives

$$\begin{aligned}
&a_{\phi,\theta}(y_{n+1}, t_{n+1}) - a_{\phi,\theta}(z_{n+1}, t_{n+1}) \\
&\quad = A_\phi(t_{n+1})(y_{n+1} - z_{n+1}) \\
&\qquad + B_\phi(t_{n+1}) \left[v_\theta(y_{n+1}, t_{n+1}) - v_\theta(z_{n+1}, t_{n+1}) \right] \\
&\quad = A_\phi(t_{n+1}) e_{n+1} + B_\phi(t_{n+1}) \left[v_\theta(y_{n+1}, t_{n+1}) - v_\theta(z_{n+1}, t_{n+1}) \right].
\end{aligned}$$

Substituting this identity into the preceding expression yields

$$\begin{aligned}
e_n = e_{n+1} &- \Delta_n \left[v_\theta(y_{n+1}, t_{n+1}) - v^*(z_{n+1}, t_{n+1}) \right] \\
&+ \frac{\Delta_n^2}{2} A_\phi(t_{n+1}) e_{n+1} \\
&+ \frac{\Delta_n^2}{2} B_\phi(t_{n+1}) \left[v_\theta(y_{n+1}, t_{n+1}) - v_\theta(z_{n+1}, t_{n+1}) \right] - r_n.
\end{aligned}$$

To separate propagation error from velocity-model error, add and subtract $v_\theta(z_{n+1}, t_{n+1})$:

$$\begin{aligned}
&v_\theta(y_{n+1}, t_{n+1}) - v^*(z_{n+1}, t_{n+1}) \\
&\quad = \left[v_\theta(y_{n+1}, t_{n+1}) - v_\theta(z_{n+1}, t_{n+1}) \right] + \left[v_\theta(z_{n+1}, t_{n+1}) - v^*(z_{n+1}, t_{n+1}) \right].
\end{aligned}$$

Substituting this equality above and collecting the two occurrences of $v_\theta(y_{n+1}, t_{n+1}) - v_\theta(z_{n+1}, t_{n+1})$ yields

$$\begin{aligned}
e_n = &\left(1 + \frac{\Delta_n^2}{2} A_\phi(t_{n+1}) \right) e_{n+1} \\
&- \left(\Delta_n - \frac{\Delta_n^2}{2} B_\phi(t_{n+1}) \right) \left[v_\theta(y_{n+1}, t_{n+1}) - v_\theta(z_{n+1}, t_{n+1}) \right] \\
&- \Delta_n \left[v_\theta(z_{n+1}, t_{n+1}) - v^*(z_{n+1}, t_{n+1}) \right] - r_n.
\end{aligned}$$

The four terms respectively represent propagation of the existing state error, the effect of evaluating v_θ at the perturbed numerical state, velocity-model error along the exact trajectory, and the local acceleration remainder. Taking norms and applying the triangle inequality yields

$$\begin{aligned}
\|e_n\| \le &\left| 1 + \frac{\Delta_n^2}{2} A_\phi(t_{n+1}) \right| \|e_{n+1}\| \\
&+ \left| \Delta_n - \frac{\Delta_n^2}{2} B_\phi(t_{n+1}) \right| \|v_\theta(y_{n+1}, t_{n+1}) - v_\theta(z_{n+1}, t_{n+1})\| \\
&+ \Delta_n \varepsilon_n^{\text{net}} + \|r_n\|.
\end{aligned} \tag{26}$$

The uniform spatial Lipschitz assumption gives

$$\|v_\theta(y_{n+1}, t_{n+1}) - v_\theta(z_{n+1}, t_{n+1})\| \le L\|e_{n+1}\|.$$

Moreover,

$$\left|1 + \frac{\Delta_n^2}{2} A_\phi(t_{n+1})\right| \le 1 + \frac{\Delta_n^2}{2}|A_\phi(t_{n+1})|,$$

and

$$\left|\Delta_n - \frac{\Delta_n^2}{2} B_\phi(t_{n+1})\right| \le \Delta_n + \frac{\Delta_n^2}{2}|B_\phi(t_{n+1})|.$$

Substituting the three preceding estimates and (25) into (26) gives

$$\begin{aligned}\|e_n\| \le & \left[1 + \frac{\Delta_n^2}{2}|A_\phi(t_{n+1})| + L\Delta_n + \frac{L\Delta_n^2}{2}|B_\phi(t_{n+1})|\right]\|e_{n+1}\| \\ & + \Delta_n \varepsilon_n^{\text{net}} + \frac{\Delta_n^2}{2}\|R^*_{\phi,\theta}(z_{n+1}, t_{n+1})\| + C\Delta_n^3.\end{aligned}$$

The coefficient of $\|e_{n+1}\|$ can be regrouped as

$$\begin{aligned}& 1 + L\Delta_n + \frac{\Delta_n^2}{2}\left(|A_\phi(t_{n+1})| + L|B_\phi(t_{n+1})|\right) \\ & \le 1 + L\Delta_n + \frac{\Delta_n^2}{2}\left(M_A + LM_B\right) \\ & = \gamma_n + K_{\text{coef}}\Delta_n^2,\end{aligned}$$

where the inequality uses the fixed bounds $\|A_\phi\|_\infty \le M_A$ and $\|B_\phi\|_\infty \le M_B$, and the last equality uses the definitions of γ_n and K_{coef}. The remaining three terms are collected into β_n. Therefore,

$$\|e_n\| \le (\gamma_n + K_{\text{coef}}\Delta_n^2)\|e_{n+1}\| + \beta_n, \tag{27}$$

where

$$\beta_n := \Delta_n \varepsilon_n^{\text{net}} + \frac{\Delta_n^2}{2}\|R^*_{\phi,\theta}(z_{n+1}, t_{n+1})\| + C\Delta_n^3.$$

For every partial product,

$$\begin{aligned}\prod_{j=0}^{k-1}(\gamma_j + K_{\text{coef}}\Delta_j^2) &\le \left(\prod_{j=0}^{k-1}\gamma_j\right)\exp\left(K_{\text{coef}}\sum_{j=0}^{k-1}\Delta_j^2\right) \\ &\le e^{K_{\text{coef}}\Delta_{\max}T}\Lambda_k.\end{aligned}$$

Iterating (27) backward therefore yields the sharper pathwise estimate

$$\|e_0\| \le e^{K_{\text{coef}}\Delta_{\max}T}\left[\sum_{k=0}^{N-1}\Lambda_k\beta_k\right]. \tag{28}$$

Here we used the shared-terminal initialization in Theorem 2, which gives $e_N = y_N - Z_{t_N} = 0$ almost surely. Thus, every error in (28) is generated by one of the local defects β_k.

Step 2: Expected bound with the exact residual. Draw $Z_{t_N} \sim p_{\text{ref}}$ and let $Z_{t_n} := Z(t_n)$ be the exact reverse trajectory generated from this terminal sample. Initialize the numerical trajectory with the same sample, $y_N = Z_{t_N}$, and generate it by (8). Let $\mathbb{P}_{\text{rev}}$ denote the resulting joint distribution of $(Z_{t_0}, \dots, Z_{t_N}, y_0, \dots, y_N)$. The output error $e_0 = y_0 - Z_{t_0}$ depends on both trajectories, so its expectation is taken under this joint distribution. For brevity, for any integrable function of the exact state at time t_{k+1}, write

$$\mathbb{E}_k[\,\cdot\,] := \mathbb{E}_{Z_{t_{k+1}} \sim p_{t_{k+1}}(\cdot|s)}[\,\cdot\,], \qquad 0 \le k \le N-1.$$

Thus, $\mathbb{E}_k$ averages only over the marginal exact state $Z_{t_{k+1}}$. The numerical trajectory is absent from these local terms and is therefore marginalized out.

To take expectations of the local defects in (28), observe that, for each k, both

$$\mathcal{E}_k^{\text{net}} = \|v_\theta(Z_{t_{k+1}}, t_{k+1}) - v^*(Z_{t_{k+1}}, t_{k+1})\|$$

and

$$\|R^*_{\phi,\theta}(Z_{t_{k+1}}, t_{k+1})\|$$

are functions only of the exact state $Z_{t_{k+1}}$. Since the exact reverse flow transports the marginals, $Z_{t_{k+1}} \sim p_{t_{k+1}}(\cdot \mid s)$. Therefore, for any integrable function g_k of this state,

$$\begin{aligned}\mathbb{E}_{\mathbb{P}_{\text{rev}}}[g_k(Z_{t_{k+1}})] &= \int g_k(z)\, \mathbb{P}_{\text{rev}}(dZ, dy) \\ &= \int g_k(z)\, p_{t_{k+1}}(dz \mid s) = \mathbb{E}_k[g_k(Z_{t_{k+1}})].\end{aligned}$$

Taking expectations termwise in (28) gives

$$\mathbb{E}_{\mathbb{P}_{\text{rev}}}\|e_0\| \le e^{K_{\text{coef}}\Delta_{\max}T}\left[\sum_{k=0}^{N-1}\Lambda_k\left(\Delta_k\mathbb{E}_k[\mathcal{E}_k^{\text{net}}] + \frac{\Delta_k^2}{2}\mathbb{E}_k\|R^*_{\phi,\theta}(Z_{t_{k+1}}, t_{k+1})\| + C\Delta_k^3\right)\right].$$

We now bound each nonresidual contribution,

$$\Lambda_k = \prod_{j=0}^{k-1}(1 + L\Delta_j) \le \exp\left(L\sum_{j=0}^{k-1}\Delta_j\right) \le e^{LT}. \tag{29}$$

Also, $\Delta_{\max} \le T$, so

$$e^{K_{\text{coef}}\Delta_{\max}T} \le e^{K_{\text{coef}}T^2}. \tag{30}$$

For the learned-field error, first bound each Λ_k and each expected error by their respective maxima, and then use $\sum_k \Delta_k = T$. Together with (29) and (30), this gives

$$\begin{aligned}&e^{K_{\text{coef}}\Delta_{\max}T}\sum_{k=0}^{N-1}\Lambda_k\Delta_k\mathbb{E}_k[\mathcal{E}_k^{\text{net}}] \\ &\quad\le e^{K_{\text{coef}}T^2}\left(\max_{0\le j\le N-1}\Lambda_j\right)\left(\sum_{k=0}^{N-1}\Delta_k\right)\max_{0\le j\le N-1}\mathbb{E}_j[\mathcal{E}_j^{\text{net}}] \\ &\quad\le e^{K_{\text{coef}}T^2+LT}T\max_{0\le k\le N-1}\mathbb{E}_k[\mathcal{E}_k^{\text{net}}].\end{aligned} \tag{31}$$

For the Taylor remainder, $\Delta_k^3 \le \Delta_{\max}^2\Delta_k$, and hence

$$e^{K_{\text{coef}}\Delta_{\max}T}C\sum_{k=0}^{N-1}\Lambda_k\Delta_k^3 \le Ce^{K_{\text{coef}}T^2+LT}\Delta_{\max}^2T. \tag{32}$$

The learned-field and Taylor-remainder estimates can be absorbed into a constant C_T depending only on T, L, C, and uniform bounds on A_ϕ and B_ϕ. For example, one may take

$$C_T := e^{K_{\text{coef}}T^2+LT}\max\{T, CT\}.$$

Combining these two estimates while leaving the exact-residual contribution in its summed form, Step 2 yields

$$\begin{aligned}\mathbb{E}_{\mathbb{P}_{\text{rev}}}\|e_0\| \le C_T&\left(\max_{0\le k\le N-1}\mathbb{E}_k[\mathcal{E}_k^{\text{net}}] + \Delta_{\max}^2\right) \\ &+ e^{K_{\text{coef}}\Delta_{\max}T}\sum_{k=0}^{N-1}\Lambda_k\frac{\Delta_k^2}{2}\mathbb{E}_k\|R^*_{\phi,\theta}(Z_{t_{k+1}}, t_{k+1})\|.\end{aligned} \tag{33}$$

Step 3: Replacement by the computable residual. The exact residual in (33) is not directly computable because it contains a^*. Because the exact and computable residuals use the same structured acceleration,

$$R^*_{\phi,\theta}(z,t) = R^\theta_{\phi,\theta}(z,t) + a^*(z,t) - a_\theta(z,t), \tag{34}$$

and hence

$$\|R^*_{\phi,\theta}(z,t)\| \le \|R^\theta_{\phi,\theta}(z,t)\| + \|a^*(z,t) - a_\theta(z,t)\|. \tag{35}$$

At $(Z_{t_{k+1}}, t_{k+1})$, the second term is precisely the acceleration-model error $\mathcal{E}^{\mathrm{acc}}_k$. Taking $\mathbb{E}_k$ in (35), multiplying by the nonnegative weight $e^{K_{\mathrm{coef}}\Delta_{\max}T}\Lambda_k\Delta_k^2/2$, and summing over k gives

$$\begin{aligned} &e^{K_{\mathrm{coef}}\Delta_{\max}T}\sum_{k=0}^{N-1}\Lambda_k\frac{\Delta_k^2}{2}\mathbb{E}_k\|R^*_{\phi,\theta}(Z_{t_{k+1}},t_{k+1})\| \\ &\quad\le e^{K_{\mathrm{coef}}\Delta_{\max}T}\sum_{k=0}^{N-1}\Lambda_k\frac{\Delta_k^2}{2}\mathbb{E}_k\|R^\theta_{\phi,\theta}(Z_{t_{k+1}},t_{k+1})\| + e^{K_{\mathrm{coef}}\Delta_{\max}T}\sum_{k=0}^{N-1}\Lambda_k\frac{\Delta_k^2}{2}\mathbb{E}_k[\mathcal{E}^{\mathrm{acc}}_k]. \end{aligned}$$

The first sum is computable and contains the dependence on the coefficient head. The second sum measures the discrepancy between a_θ and a^*; it depends only on the frozen learned field and the population field, and is independent of (A_ϕ, B_ϕ). We bound these two sums separately below.

For the acceleration-model contribution, use $\Lambda_k \le e^{LT}$ and

$$\sum_{k=0}^{N-1}\Delta_k^2 \le \Delta_{\max}\sum_{k=0}^{N-1}\Delta_k = \Delta_{\max}T.$$

Then

$$e^{K_{\mathrm{coef}}\Delta_{\max}T}\sum_{k=0}^{N-1}\Lambda_k\frac{\Delta_k^2}{2}\mathbb{E}_k[\mathcal{E}^{\mathrm{acc}}_k] \le \frac{T}{2}e^{K_{\mathrm{coef}}T^2+LT}\Delta_{\max}\max_{0\le k\le N-1}\mathbb{E}_k[\mathcal{E}^{\mathrm{acc}}_k]. \tag{36}$$

Enlarging C_T, if necessary, absorbs this contribution into $C_T\Delta_{\max}\max_k\mathbb{E}_k[\mathcal{E}^{\mathrm{acc}}_k]$, the acceleration-model component of $C_T\mathcal{B}_\theta(\mathcal{T})$.

For the computable residual contribution, the sum contains N nonnegative terms. Bounding it by N times its largest term and using (30) gives

$$\begin{aligned} &e^{K_{\mathrm{coef}}\Delta_{\max}T}\sum_{k=0}^{N-1}\Lambda_k\frac{\Delta_k^2}{2}\mathbb{E}_k\|R^\theta_{\phi,\theta}(Z_{t_{k+1}},t_{k+1})\| \\ &\quad\le \frac{N}{2}e^{K_{\mathrm{coef}}T^2}\max_{0\le k\le N-1}\Lambda_k\Delta_k^2\mathbb{E}_k\|R^\theta_{\phi,\theta}(Z_{t_{k+1}},t_{k+1})\|. \end{aligned}$$

Set $C_{T,N} := \frac{N}{2}e^{K_{\mathrm{coef}}T^2}$. By definition, the maximum on the right-hand side is $\mathcal{J}_\theta(\phi,\mathcal{T})$. The learned-field estimate (31), acceleration estimate (36), and Taylor estimate (32) together give

$$C_T\left(\max_k\mathbb{E}_k[\mathcal{E}^{\mathrm{net}}_k] + \Delta_{\max}\max_k\mathbb{E}_k[\mathcal{E}^{\mathrm{acc}}_k] + \Delta^2_{\max}\right) = C_T\mathcal{B}_\theta(\mathcal{T}).$$

Combining these bounds proves

$$\mathbb{E}_{\mathbb{P}_{\mathrm{rev}}}\|e_0\| \le C_T\mathcal{B}_\theta(\mathcal{T}) + C_{T,N}\mathcal{J}_\theta(\phi,\mathcal{T}),$$

which is (11). The constants depend only on T, the regularity constants, the fixed bounds M_A, M_B, and, for $C_{T,N}$, the step budget N. In particular, they are independent of ϕ, the grid locations, and the residual values.

A.5 Coefficient-head design for unified guided sampling

The coefficient head is trained after the pretrained velocity field has been fixed. It takes the time, condition, and guidance weight as inputs, each represented by an embedding:

$$\eta(t,s,w) := [e_t(t), e_s(s), e_w(w)]\,.$$

Here e_t, e_s, and e_w denote the time, condition, and guidance embeddings, respectively. The guidance-aware head is then

$$\big(A_\phi(t,s,w), B_\phi(t,s,w)\big) = h_\phi(\eta(t,s,w))\,.$$

For unconditional generation, $w = 0$ and $s = \emptyset$, so $e_s(s)$ is the null-condition embedding. For ordinary conditional generation, $w = 1$ and s is the given condition. For CFG, s remains the given condition and w specifies the guidance strength. Thus, one head covers the unconditional, conditional, and guided regimes. The output dimension is only two, so h_ϕ can be implemented as a small multilayer perceptron.

The condition embedding depends on the pretrained architecture. For a class-conditional model, it can be the class embedding,

$$e_s(s) = e_{\text{class}}(s), \qquad e_s(\emptyset) = e_{\text{null}}.$$

For a text-conditional model, $e_s(s)$ can reuse an internal text conditioning representation when the model exposes one. Otherwise, it can be formed by pooling the text embedding, for example

$$e_s(s) = \text{Pool}(e_{\text{text}}(s)),$$

with the null-prompt embedding used when $s = \emptyset$. The guidance embedding $e_w(w)$ may be produced by a small learned embedding layer or a standard scalar embedding. The head can reuse these representations from the pretrained model, so it adds little overhead.

The pretrained parameters θ remain frozen during head fitting. JVPs are used only to construct the acceleration target $a_{\theta,w}$; gradients update ϕ alone. To satisfy the uniform bounds assumed in Theorem 2, the two outputs may be bounded explicitly, for example by parameterizing

$$A_\phi = M_A \tanh(\widetilde{A}_\phi), \qquad B_\phi = M_B \tanh(\widetilde{B}_\phi).$$

Once trained, the head requires neither derivatives nor an additional velocity-field evaluation at sampling time.

A.6 Efficient evaluation of the CFG acceleration

Directly evaluating (13) for every sampled guidance scale would require a separate JVP for each value of w. Instead, the dependence on w can be isolated using four branch-level JVPs. Define the guidance direction

$$d_\theta(z,t,s) := v^{\text{c}}_\theta(z,t,s) - v^{\text{u}}_\theta(z,t),$$

so that, for a time-independent guidance scale,

$$v_{\theta,w}(z,t,s) = v^{\text{u}}_\theta(z,t) + w d_\theta(z,t,s).$$

The four required branch-level JVPs are

$$\begin{aligned}
Q_{\text{u0}} &:= \text{JVP}_{(z,t)}\, v^{\text{u}}_\theta(z,t)[v^{\text{u}}_\theta, 1] = D_z v^{\text{u}}_\theta v^{\text{u}}_\theta + \partial_t v^{\text{u}}_\theta,\\
Q_{\text{c0}} &:= \text{JVP}_{(z,t)}\, v^{\text{c}}_\theta(z,t,s)[v^{\text{u}}_\theta, 1] = D_z v^{\text{c}}_\theta v^{\text{u}}_\theta + \partial_t v^{\text{c}}_\theta,\\
Q_{\text{u1}} &:= \text{JVP}_{(z,t)}\, v^{\text{u}}_\theta(z,t)[d_\theta, 0] = D_z v^{\text{u}}_\theta d_\theta,\\
Q_{\text{c1}} &:= \text{JVP}_{(z,t)}\, v^{\text{c}}_\theta(z,t,s)[d_\theta, 0] = D_z v^{\text{c}}_\theta d_\theta.
\end{aligned} \tag{37}$$

All fields and derivatives in (37) are evaluated at the same (z,t,s), with s omitted from the unconditional branch. Expanding the total derivative of $v_{\theta,w} = v^{\text{u}}_\theta + w d_\theta$ then gives

$$a_{\theta,w} = Q_{\text{u0}} + w\big(Q_{\text{c0}} - Q_{\text{u0}} + Q_{\text{u1}}\big) + w^2\big(Q_{\text{c1}} - Q_{\text{u1}}\big). \tag{38}$$

Thus, each training tuple requires four branch-level JVP evaluations. Their outputs can be reused to construct the exact guided acceleration for any number of sampled guidance scales, avoiding a separate JVP evaluation for each value of w.

Offline collection, head fitting, and profile construction. We separate derivative-based data collection from coefficient-head training. We first draw a fixed pool of M_{base} triplets $\{(X_0^{(j)}, \xi^{(j)}, S^{(j)})\}_{j=1}^{M_{\mathrm{base}}}$. To form each derivative record, we independently sample $t \sim \mathrm{Unif}[0,T]$ and a pool index $J \sim \mathrm{Unif}\{1,\dots,M_{\mathrm{base}}\}$, so the same base triplets are reused across different times. The resulting state, branch velocities, and four JVPs are written to sharded files. Head training streams minibatches from this fixed on-disk dataset, requiring no further base-model or JVP evaluations. After fitting, let $I_r = (\tau_{r-1}, \tau_r]$, $r = 1,\dots,K$, denote bins on a fine grid, with the first bin also including $t = 0$, and let $\mathcal{D}_r := \{m : t^{(m)} \in I_r\}$. We implement the empirical profile in (15) by the binned estimator

$$\widehat{R}_w(\tau_r) = \frac{1}{|\mathcal{D}_r|} \sum_{m \in \mathcal{D}_r} \left\| a_{\theta,w}^{(m)} - A_\phi(t^{(m)}, S^{(m)}, w) X_t^{(m)} - B_\phi(t^{(m)}, S^{(m)}, w) v_{\theta,w}^{(m)} \right\|,$$

where $v_{\theta,w}^{(m)}$ and $a_{\theta,w}^{(m)}$ are reconstructed from the stored branch quantities. In practice, the profile is accumulated in one streaming pass: each record is assigned to its bin once, rather than searching the stored files separately for every bin. Algorithm 1 summarizes the complete pipeline.

Algorithm 1 Offline data collection, head fitting, and residual profiling

Require: Frozen branches $v_\theta^{\mathrm{u}}, v_\theta^{\mathrm{c}}$, base-pool size M_{base}, record count M, guidance distribution p_W, bins $\{I_r\}_{r=1}^K$

Phase I: continuous-time data collection

Draw and fix $\mathcal{P} = \{(X_0^{(j)}, \xi^{(j)}, S^{(j)})\}_{j=1}^{M_{\mathrm{base}}}$
for $m = 1,\dots,M$ **do**
  Sample $t^{(m)} \sim \mathrm{Unif}[0,T]$ and $J_m \sim \mathrm{Unif}\{1,\dots,M_{\mathrm{base}}\}$
  Construct $X_t^{(m)} = \alpha(t^{(m)}) X_0^{(J_m)} + \sigma(t^{(m)}) \xi^{(J_m)}$, and set $S^{(m)} = S^{(J_m)}$
  Evaluate $v_{\mathrm{u}}^{(m)}, v_{\mathrm{c}}^{(m)}$ and $d^{(m)} = v_{\mathrm{c}}^{(m)} - v_{\mathrm{u}}^{(m)}$
  Evaluate and store $Q_{\mathrm{u0}}^{(m)}, Q_{\mathrm{c0}}^{(m)}, Q_{\mathrm{u1}}^{(m)}, Q_{\mathrm{c1}}^{(m)}$ using (37)
  Write the record $\mathcal{C}_m = (t^{(m)}, S^{(m)}, X_t^{(m)}, v_{\mathrm{u}}^{(m)}, d^{(m)}, Q_{\mathrm{u0}}^{(m)}, Q_{\mathrm{c0}}^{(m)}, Q_{\mathrm{u1}}^{(m)}, Q_{\mathrm{c1}}^{(m)})$ to a sharded data file
end for

Phase II: coefficient-head training from stored files

while the head has not converged **do**
  Load a minibatch $\mathcal{B}$ from the stored files and sample $W_m \sim p_W$ for $m \in \mathcal{B}$
  Reconstruct $v_{\theta,W_m}^{(m)}$ and $a_{\theta,W_m}^{(m)}$ using (12) and (38)
  Update ϕ using the minibatch version of (14)
end while

Phase III: fine-grid residual profiling

Initialize counts $n_r \leftarrow 0$ and residual sums $H_{w,r} \leftarrow 0$ for all target w and $r = 1,\dots,K$
for each stored record $m = 1,\dots,M$ **do**
  Assign it once to the unique bin r_m satisfying $t^{(m)} \in I_{r_m}$
  $n_{r_m} \leftarrow n_{r_m} + 1$
  for each target guidance scale w **do**
    Reconstruct $v_{\theta,w}^{(m)}$ and $a_{\theta,w}^{(m)}$ from the stored record
    $H_{w,r_m} \leftarrow H_{w,r_m} + \| a_{\theta,w}^{(m)} - A_\phi(t^{(m)}, S^{(m)}, w) X_t^{(m)} - B_\phi(t^{(m)}, S^{(m)}, w) v_{\theta,w}^{(m)} \|$
  end for
end for
for each target w and bin r **do**
  $\widehat{R}_w(\tau_r) \leftarrow H_{w,r}/n_r$
end for
return fitted head ϕ and profiles $\{\widehat{R}_w(\tau_r)\}_{w,r}$

A.7 Dynamic-programming formulation of timestep allocation

Fix the guidance scale w throughout. We solve the minimax schedule problem (16) on the candidate grid $0 = \tau_0 < \cdots < \tau_K = T$. Recall that $c_w(k, \ell)$ is the residual cost of taking one step from τ_ℓ to τ_k, while $\Gamma(k, \ell)$ propagates costs incurred on later intervals.

Although the candidate grid is indexed in increasing time, the sampler traverses a selected grid in reverse, from T to 0. Consequently, a defect generated on a higher-time interval is propagated through each lower-time interval traversed afterward. The recursion below is organized in increasing grid indices only to construct the schedule; its propagation factors follow this reverse-time sampling order.

Let $M_{i,k}^{(w)}$ denote the smallest achievable worst propagated cost from τ_k to T using exactly i intervals, with propagation measured relative to τ_k. The boundary conditions are

$$M_{0,K}^{(w)} = 0, \qquad M_{0,k}^{(w)} = +\infty \quad \text{for } k < K.$$

For $i \geq 1$, choosing τ_ℓ as the next grid point incurs the current interval cost $c_w(k, \ell)$. Costs on the higher-time intervals, which are encountered earlier during reverse sampling, are summarized by $M_{i-1,\ell}^{(w)}$ and propagated through the lower-time interval $[\tau_k, \tau_\ell]$ by $\Gamma(k, \ell)$. Therefore, the dynamic-programming recursion is

$$M_{i,k}^{(w)} = \min_{\ell > k} \max\Big\{ c_w(k, \ell), \Gamma(k, \ell) M_{i-1,\ell}^{(w)} \Big\}.$$

Algorithm 2 states the feasible index ranges explicitly and includes backtracking.

Algorithm 2 Residual-aware timestep allocation

Require: Costs $c_w(k, \ell)$, factors $\Gamma(k, \ell)$, step budget N, candidate-grid size K

Set all entries of M to $+\infty$; set $M_{0,K}^{(w)} \leftarrow 0$
for $i = 1, \ldots, N$ **do**
  for $k = 0, \ldots, K - i$ **do**
    $\ell_{i,k}^* \leftarrow \arg\min_{k < \ell \leq K - i + 1} \max\Big\{ c_w(k, \ell), \Gamma(k, \ell) M_{i-1,\ell}^{(w)} \Big\}$
    $M_{i,k}^{(w)} \leftarrow \max\Big\{ c_w(k, \ell_{i,k}^*), \Gamma(k, \ell_{i,k}^*) M_{i-1,\ell_{i,k}^*}^{(w)} \Big\}$
  end for
end for
$k_0 \leftarrow 0$
for $r = 0, \ldots, N - 1$ **do**
  $k_{r+1} \leftarrow \ell_{N-r,k_r}^*$
end for
return schedule $\{\tau_{k_r}\}_{r=0}^{N}$ and value $M_{N,0}^{(w)}$

Using prefix sums to form the quadrature averages, all interval costs are precomputed in $\Theta(K)$ time. The dynamic program takes $\Theta(NK^2)$ time and $\Theta(NK)$ memory, including the stored successors.

A.8 Second-order accuracy of structured-predictor Heun

Proposition 3 (Second-order accuracy of structured-predictor Heun)**.** *Fix (s, w). Suppose that $v_{\theta,w}$ is twice continuously differentiable and that $a_{\phi,\theta,w}$ is locally Lipschitz. Then* (17)– (18) *has local truncation error $O(\Delta_n^3)$ for $\dot{z} = v_{\theta,w}(z, t, s)$. On any finite interval where the exact solution remains in a compact subset of the domain, the method is globally second-order for sufficiently small $\Delta_{\max}$.*

Proof. Fix (s, w) and suppress them from the notation. Write $v := v_{\theta,w}$, $a_\phi := a_{\phi,\theta,w}$, and

$$a_\theta(z, t) := \partial_t v(z, t) + D_z v(z, t) v(z, t).$$

Initialize one step at an exact state $y_{n+1} = z(t_{n+1})$, and abbreviate $z := z(t_{n+1})$, $t := t_{n+1}$, and $\Delta := \Delta_n$. The exact solution of the learned guided ODE satisfies

$$z(t - \Delta) = z - \Delta v(z, t) + \frac{\Delta^2}{2} a_\theta(z, t) + O(\Delta^3). \tag{39}$$

The structured predictor is

$$\widetilde{y} = z - \Delta v(z,t) + \frac{\Delta^2}{2} a_\phi(z,t) = z - \Delta v(z,t) + O(\Delta^2).$$

Because a_ϕ is locally Lipschitz, it is locally bounded, and the structured term is $O(\Delta^2)$. Hence $\widetilde{y} - z = -\Delta v(z,t) + O(\Delta^2)$ and $\|\widetilde{y} - z\| = O(\Delta)$. Thus, the structured correction does not change the first-order displacement of the predictor.

A joint space–time Taylor expansion of the velocity at $(\widetilde{y}, t - \Delta)$ has a second-order remainder because both the state and time displacements are $O(\Delta)$. Therefore,

$$\begin{aligned} v(\widetilde{y}, t-\Delta) &= v(z,t) + D_z v(z,t)(\widetilde{y} - z) - \Delta \partial_t v(z,t) + O(\Delta^2) \\ &= v(z,t) - \Delta D_z v(z,t) v(z,t) - \Delta \partial_t v(z,t) + O(\Delta^2) \\ &= v(z,t) - \Delta a_\theta(z,t) + O(\Delta^2). \end{aligned} \tag{40}$$

The $O(\Delta^2)$ structured predictor correction contributes only to the $O(\Delta^2)$ remainder in this velocity evaluation.

Substituting (40) into the Heun correction (18) yields

$$\begin{aligned} y_n &= z - \frac{\Delta}{2}\left[v(z,t) + v(\widetilde{y}, t-\Delta)\right] \\ &= z - \frac{\Delta}{2}\left[2v(z,t) - \Delta a_\theta(z,t) + O(\Delta^2)\right] \\ &= z - \Delta v(z,t) + \frac{\Delta^2}{2} a_\theta(z,t) + O(\Delta^3). \end{aligned}$$

Comparison with (39) gives local truncation error $O(\Delta^3)$. On a compact neighborhood of the exact trajectory, v and a_ϕ are Lipschitz in the state, so the one-step map has Lipschitz factor $1 + O(\Delta)$. For sufficiently small $\Delta_{\max}$, the numerical trajectory remains in this neighborhood, and accumulating the local errors over the stable steps gives global error $O(\Delta_{\max}^2)$. □

A.9 Additional experimental details and results

This section provides supplementary implementation details and supporting results for the CIFAR-10 and ImageNet-256 experiments.

A.9.1 CIFAR-10 VP-cosine velocity network

The CIFAR-10 VP-cosine experiment uses an unconditional image-space U-Net. The network maps a state $X_t \in \mathbb{R}^{3\times32\times32}$ and time to a velocity prediction of the same shape. It has a multiresolution encoder–decoder with skip connections, residual blocks, and self-attention, and uses a sinusoidal time embedding processed by an MLP to condition the blocks.

Training follows the VP-cosine interpolant

$$X_t = \cos(t) X_0 + \sin(t)\xi, \qquad t \in [0, \pi/2].$$

The implementation target $\cos(t)\xi - \sin(t)X_0$ is the velocity $\dot{X}_t$. Consequently, the analytic backbone is $\ddot{X}_t = -X_t$.

To construct Figure 1, we draw one fixed set of 5,000 endpoint pairs $\{(X_0^{(m)}, \xi^{(m)})\}_{m=1}^{5000}$ and reuse it at every evaluated time. For each resulting $X_t^{(m)}$, we compute $a_\theta = \partial_t v_\theta + D_z v_\theta v_\theta$ via a JVP and evaluate the Euler and backbone residual norms $\|a_\theta\|$ and $\|a_\theta + X_t\|$. Evaluating both residuals on the same samples gives the paired-difference distribution shown in the middle panel. We compute CIFAR-10 FID and recall using TorchFidelity (Obukhov et al., 2020) with 1,024-dimensional features from the global average pooling layer of Inception-v1. Full architectural settings, the implementation (`model_architecture.py`), the trained checkpoint, and the residual-profile code are provided in the supplementary material.

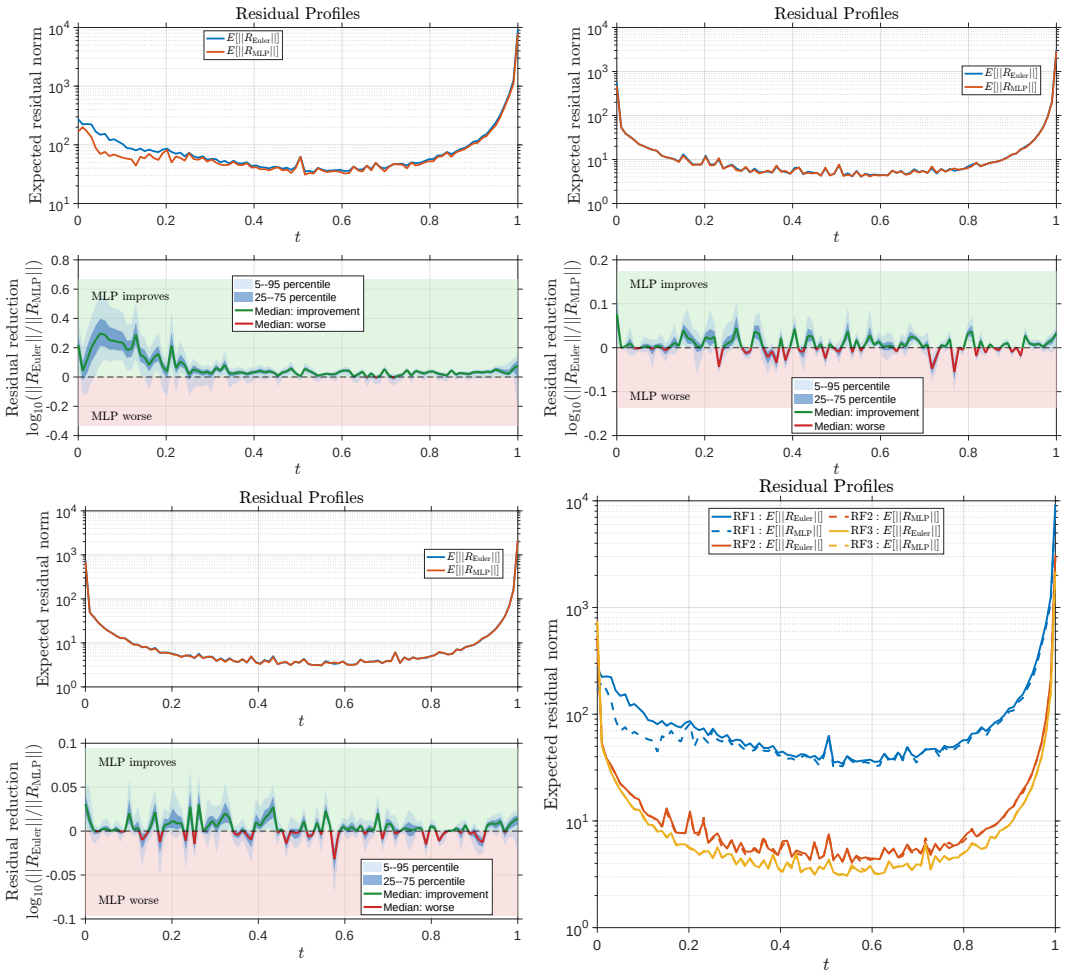

Figure 2: CIFAR-10 unconditional Rectified Flow residual diagnostics. Top left: 1-RF residual profiles and paired residual reduction. Top right: the corresponding 2-RF results. Bottom left: the corresponding 3-RF results. Bottom right: comparison of the residual profiles across 1-RF, 2-RF, and 3-RF. Residual statistics use $\Delta t = 0.01$ and 5,000 paired samples per time. Following the Rectified Flow convention (Liu et al., 2023), $t = 0$ corresponds to the noise endpoint and $t = 1$ to the data endpoint.

A.9.2 CIFAR-10 Rectified Flow models

We use the pretrained CIFAR-10 1-RF, 2-RF, and 3-RF checkpoints released with Rectified Flow (Liu et al., 2023) (`https://github.com/gnobitab/RectifiedFlow`). The pretrained vector fields remain frozen when fitting the lightweight coefficient heads and throughout sampling. The experiment scripts are provided in the supplementary material.

The 512-dimensional input to each coefficient head is the output of the frozen velocity network's timestep-embedding MLP, namely the `temb` vector supplied to its ResNet blocks. The coefficient heads fitted for 1-RF, 2-RF, and 3-RF use the same MLP architecture:

$$\mathbb{R}^{512} \xrightarrow{\text{Linear}(512,256)} \xrightarrow{\text{SiLU}} \xrightarrow{\text{Linear}(256,256)} \xrightarrow{\text{SiLU}} \xrightarrow{\text{Linear}(256,2)} \mathbb{R}^2.$$

All linear layers include biases. The two outputs parameterize A_ϕ and B_ϕ, and each coefficient head contains 197,634 trainable parameters. By comparison, each frozen RF velocity network contains more than 61.8 million parameters, so the coefficient head adds approximately 0.32% as many parameters. Each coefficient MLP is trained using 250,000 random forward-interpolant samples, with t sampled uniformly from $[0, 1)$.

Figure 2 shows that the fitted structural residual is smaller than the Euler residual most prominently for 1-RF. The reduction becomes more modest for 2-RF and smaller still for 3-RF. This trend is explained by the reflow procedure, which progressively straightens the transport trajectories and thereby reduces the magnitude of the vector-field acceleration. Because the Euler residual is precisely this acceleration, its profile decreases across successive rectifications, leaving less acceleration for the structured coefficient head to capture. The combined profiles in the bottom-right panel make this progressive reduction clear: both the overall acceleration scale and the additional reduction supplied by the structured approximation diminish from 1-RF to 3-RF. Figure 3 shows the corresponding fitted coefficient functions. For readability, the vertical coordinates use signed-log spacing, $y_{\text{plot}} = \text{sign}(y)\log_{10}(1 + 0.1|y|)$, while the tick labels report the original values of A_ϕ and B_ϕ. The 1-RF coefficients have the largest magnitude and strongest time dependence, whereas the 2-RF and 3-RF coefficients remain much closer to zero over most of the trajectory. This behavior is consistent with successive reflow reducing the acceleration structure that the coefficient head must capture.

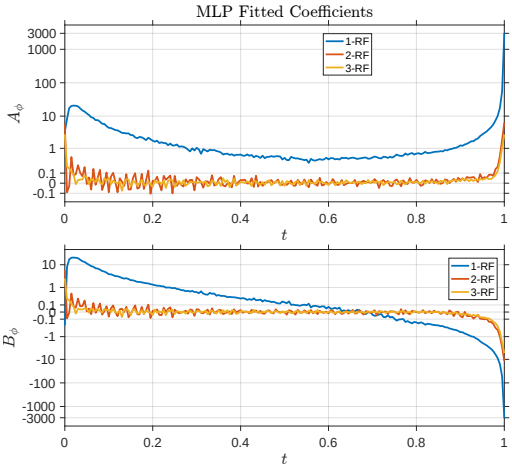

Figure 3: Fitted coefficient profiles for the CIFAR-10 unconditional Rectified Flow models. Top: $A_\phi(t)$. Bottom: $B_\phi(t)$. Curves are shown for 1-RF, 2-RF, and 3-RF.

Selection of the propagation parameter. For the CIFAR-10 unconditional 1-RF experiments, we sweep $L \in \{1.0, 1.5, 2.0, 2.5, 3.0\}$ for every sampler in Table 1 that uses a DPA or DPR schedule, evaluating each configuration with 5,000-sample FID at every NFE budget. At $\text{NFE} = 4$, $L = 2.0$ consistently achieves the lowest FID by a sizeable margin. At $\text{NFE} \in \{8, 16, 32, 64, 128, 256, 512\}$, $L = 1.0$ achieves the lowest FID in all cases except RFS-DPA at $\text{NFE} \in \{16, 128, 256, 512\}$. In these four cases, the FID at $L = 1$ exceeds the minimum over the sweep by only 0.97%, 0.01%, 0.13%, and 0.12%, respectively. Given these small differences, we adopt a single rule across all tested samplers and both schedules: $L = 2$ for $\text{NFE} = 4$ and $L = 1$ for $\text{NFE} \in \{8, 16, 32, 64, 128, 256, 512\}$. This coarse sweep is intended to identify a simple shared choice of L, rather than to optimize it exhaustively for each sampler and NFE budget. Figure 4 compares the DPA and DPR timestep allocations with the uniform schedule. The bottom panel shows how replacing the acceleration profile with the learned residual profile shifts timestep locations. The predominantly positive differences over early and middle step indices indicate that DPR reaches larger times at the same step index, toward the data endpoint under the RF convention. This is consistent with the stronger residual reduction near the noise endpoint in Figure 2, which permits coarser steps in that region. The small negative excursions near the end for some budgets indicate shifts toward smaller times. The difference measures timestep displacement rather than a local step-size change: writing $d_k = t_k^{\text{DPR}} - t_k^{\text{DPA}}$ and $h_k = t_{k+1} - t_k$, we have $h_k^{\text{DPR}} - h_k^{\text{DPA}} = d_{k+1} - d_k$. Thus, a rising difference indicates larger DPR steps locally, a falling difference indicates smaller steps, and a plateau indicates similar step sizes despite shifted timestep locations.

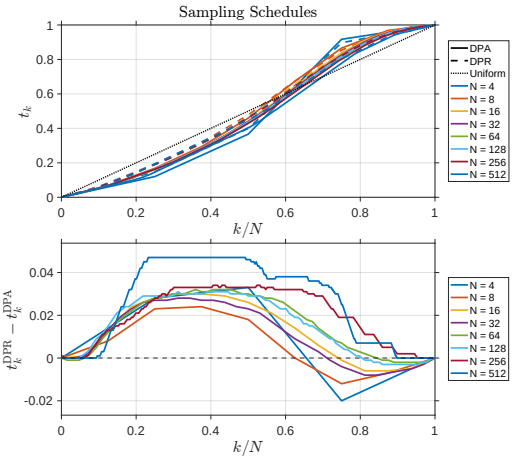

Figure 4: Sampling schedules for CIFAR-10 unconditional 1-RF generation. Top: timestep locations t_k versus normalized step index k/N, where N is the number of sampling steps. Solid and dashed curves show DPA and DPR, respectively; the dotted diagonal shows the uniform schedule. Bottom: the pointwise difference $t_k^{\text{DPR}} - t_k^{\text{DPA}}$.

Table 4: CIFAR-10 unconditional sampling with 2-RF (left) and 3-RF (right). Columns report NFE; each panel reports FID and recall (%). Methods are grouped by baseline sampler, with uniform-schedule baseline names in bold. Bold values indicate the lowest FID or highest recall within each solver block at each NFE, separately for each model, including ties at the displayed precision. Method names follow Table 1.

2-RF

Sampler-Schedule	4	8	16	32	64
	(a) FID ↓				
EUL-U	4.63	3.91	3.61	3.49	3.46
EUL-DPA	**4.27**	**3.76**	3.57	3.49	3.46
SAS-U	9.02	4.72	3.73	3.53	3.46
SAS-DPA	4.90	3.78	**3.56**	3.49	**3.45**
SAS-DPR	4.90	3.84	**3.56**	**3.48**	3.46
HEU-U	61.8	24.9	7.59	3.62	**3.37**
HEU-DPA	63.3	7.04	**3.40**	**3.43**	3.43
SPH-U	60.8	23.6	7.85	3.55	3.39
SPH-DPA	**56.1**	**6.44**	3.41	**3.43**	3.43
SPH-DPR	56.3	7.24	3.41	**3.43**	3.43
AB2-U	3.89	3.55	3.43	**3.41**	**3.42**
AB2-DPA	**3.22**	**3.35**	**3.42**	3.43	3.43
RFS-U	7.43	4.57	3.70	3.49	**3.43**
RFS-DPA	**7.19**	**3.66**	**3.44**	**3.42**	**3.43**
	(b) Recall (%) ↑				
EUL-U	49.6	51.9	53.3	53.7	53.9
EUL-DPA	50.5	52.5	53.3	53.7	54.0
SAS-U	**56.8**	**56.6**	**55.0**	**54.3**	**54.2**
SAS-DPA	54.3	53.5	53.6	53.8	54.1
SAS-DPR	54.5	53.4	53.6	53.9	54.1
HEU-U	9.40	34.6	49.8	53.6	54.1
HEU-DPA	8.52	50.0	53.9	54.1	**54.2**
SPH-U	11.4	36.9	50.4	53.9	**54.2**
SPH-DPA	**11.7**	**51.1**	**54.0**	**54.2**	**54.2**
SPH-DPR	**11.7**	50.5	**54.0**	**54.2**	**54.2**
AB2-U	52.4	53.8	54.1	**54.3**	**54.2**
AB2-DPA	**58.1**	**55.0**	**54.4**	54.2	54.1
RFS-U	**58.8**	**55.8**	54.4	54.2	**54.2**
RFS-DPA	58.4	54.6	**54.7**	**54.3**	**54.2**

3-RF

Sampler-Schedule	4	8	16	32	64
	(a) FID ↓				
EUL-U	4.46	4.15	4.00	3.94	**3.92**
EUL-DPA	4.28	4.07	3.98	3.94	3.93
SAS-U	5.13	4.26	4.03	3.94	**3.92**
SAS-DPA	4.30	4.08	**3.97**	**3.93**	**3.92**
SAS-DPR	**4.26**	**4.06**	**3.97**	**3.93**	**3.92**
HEU-U	**52.6**	18.7	5.87	**3.82**	**3.84**
HEU-DPA	55.7	18.7	5.87	**3.82**	**3.84**
SPH-U	58.5	17.4	5.57	**3.82**	3.85
SPH-DPA	73.0	**5.26**	**3.86**	3.91	3.91
SPH-DPR	55.6	5.31	**3.86**	3.91	3.91
AB2-U	4.14	3.97	3.91	**3.90**	**3.90**
AB2-DPA	**3.75**	**3.86**	**3.90**	3.91	3.91
RFS-U	7.63	4.95	4.14	3.96	**3.91**
RFS-DPA	**7.19**	**4.15**	**3.90**	**3.90**	**3.91**
	(b) Recall (%) ↑				
EUL-U	51.2	52.2	52.8	53.2	**53.3**
EUL-DPA	51.7	52.5	53.0	53.1	53.2
SAS-U	**57.6**	**55.0**	**53.8**	**53.3**	**53.3**
SAS-DPA	53.7	52.9	53.1	53.2	53.2
SAS-DPR	53.4	52.9	53.0	53.2	53.2
HEU-U	13.2	38.3	50.1	52.8	**53.3**
HEU-DPA	11.5	50.6	**53.2**	**53.3**	**53.3**
SPH-U	**13.8**	40.3	50.8	52.9	53.2
SPH-DPA	10.6	**51.1**	53.1	**53.3**	**53.3**
SPH-DPR	12.4	50.9	**53.2**	**53.3**	**53.3**
AB2-U	52.6	53.1	53.3	**53.5**	53.3
AB2-DPA	**56.3**	**53.9**	**53.5**	53.3	**53.5**
RFS-U	**60.2**	**56.0**	**53.9**	**53.4**	**53.4**
RFS-DPA	60.0	54.5	53.5	**53.4**	53.3

Sampling after successive reflow. Table 4 follows the method matrix of Table 1 for 2-RF and 3-RF. Its results are consistent with the straighter transport trajectories and much smaller acceleration profiles in Figure 2. The smaller fitted A_ϕ and B_ϕ in Figure 3 provide complementary evidence that less acceleration structure remains to be captured after reflow. Since the Euler local defect has acceleration as its leading second-order term, these diagnostics suggest less scope for improvement from structured acceleration than for 1-RF. Correspondingly, Euler already achieves FID 4.63 and 4.46 at NFE 4 for 2-RF and 3-RF, respectively, compared with 52.0 for 1-RF. Since the checkpoints define different learned vector fields, these low-NFE improvements do not imply that successive reflow uniformly improves the best attainable FID across checkpoints.

The incremental benefit of matching the schedule to the learned residual also diminishes. For 2-RF, SAS-DPA and SAS-DPR have equal FID at NFE 4 and 16, and differ by at most 0.06 across the tested budgets; for 3-RF, their FID differs by at most 0.04. This is consistent with the smaller additional residual reduction supplied by the fitted head. Nevertheless, smaller acceleration does not guarantee that its correction improves FID on every grid. For 2-RF at NFE 4, SAS-U increases FID from 4.63 to 9.02 while increasing recall from 49.6% to 56.8%, illustrating a fidelity-coverage tradeoff. For 3-RF at NFE 4, SAS-U similarly increases FID from 4.46 to 5.13 while increasing recall from 51.2% to 57.6%, showing the same fidelity-coverage tradeoff.

Timestep allocation remains useful even when the overall acceleration scale is small. At NFE 4, transferring DPA lowers AB2 FID from 3.89 to 3.22 for 2-RF and from 4.14 to 3.75 for 3-RF, while increasing recall in both cases. At NFE 8, RFS-DPA lowers FID from 4.57 to 3.66 and from 4.95

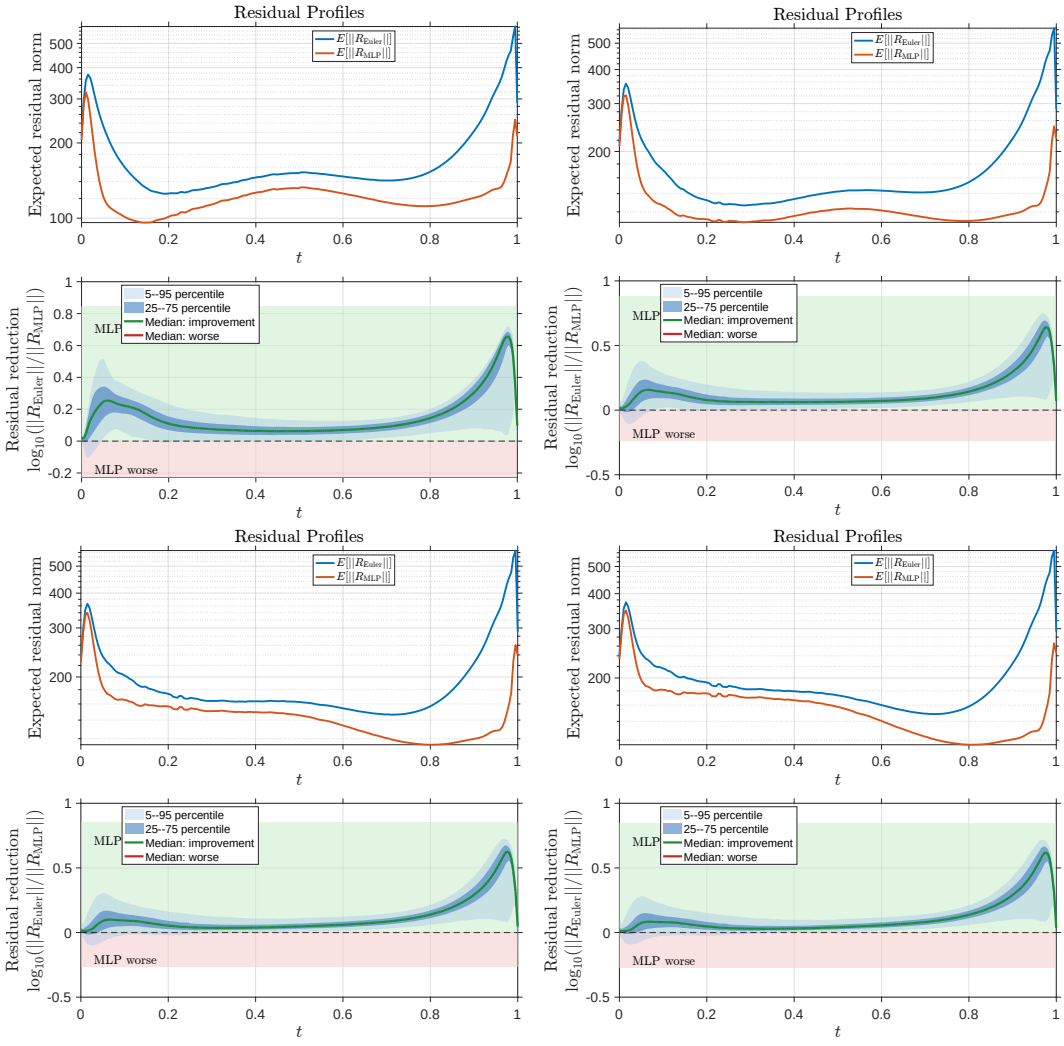

Figure 5: ImageNet-256 SiT acceleration and learned-head residual profiles. Top left: unconditional generation ($w = 0$). Top right: conditional generation without CFG ($w = 1$). Bottom left: CFG with $w = 1.375$. Bottom right: CFG with $w = 1.5$. Residual statistics use $\Delta t = 0.005$ and 5,000 paired samples per time. The endpoints $t = 0$ and $t = 1$ correspond to noise and data, respectively.

to 4.15, respectively. In the Heun block, SPH-DPA attains FID 6.44 and 5.26 at NFE 8, compared with 24.9 and 18.7 for HEU-U; this comparison changes both the predictor and the schedule, so it does not isolate the effect of timestep allocation. The AB2 and RF-Solver comparisons, however, show that scheduling alone can improve FID even for these lower-acceleration models. At NFE 32–64, many differences narrow or reverse, and DPR does not uniformly outperform DPA. Overall, the results support reduced gains from acceleration correction after reflow, while retaining solver- and budget-dependent benefits from scheduling.

A.9.3 IMAGENET-256 LATENT-SPACE EXPERIMENTS

We use the pretrained SiT-XL checkpoints released with SiT (Ma et al., 2024) (`https://github.com/willisma/SiT`) for ImageNet-256. The pretrained vector fields remain frozen when fitting the lightweight coefficient heads and throughout sampling. The experiment scripts are provided in the supplementary material.

The guidance-aware coefficient head combines time, label, and guidance embeddings. The 1,152-dimensional time and label embeddings are summed element-wise. A separate guidance embedder

Table 5: Selected propagation parameter L for ImageNet-256 sampling with SiT-XL. Each value minimizes the 3,000-sample FID over $L \in \{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ for $w = 1.375$ and $L \in \{1.0, 1.5, 2.0, 2.5, 3.0\}$ for $w \in \{1, 1.5\}$. NFE counts both CFG branches. N/E: not evaluated.

Sampler–	$w = 1$			$w = 1.375$					$w = 1.5$		
Schedule	16	64	250	12	14	16	20	30	32	128	500
SAS-DPR	3.0	3.0	3.0	N/E	N/E	N/E	N/E	N/E	2.0	1.0	1.0
SPH-DPA	1.0	2.0	N/E	0.0	N/E	0.0	0.0	N/E	1.0	2.0	N/E
SPH-DPR	1.0	1.0	1.0	N/E	N/E	N/E	N/E	N/E	1.0	1.0	1.0
AB2-DPA	3.0	3.0	N/E	3.0	3.0	3.0	3.0	2.5	1.0	3.0	N/E
RFS-DPA	3.0	3.0	N/E	2.5	N/E	3.0	3.0	N/E	1.0	1.0	N/E

maps (w, w^2) to a 32-dimensional vector using

$$\mathbb{R}^2 \xrightarrow{\text{Linear}(2,64)} \xrightarrow{\text{SiLU}} \xrightarrow{\text{Linear}(64,32)} \xrightarrow{\text{SiLU}} \mathbb{R}^{32}.$$

Concatenating this 32-dimensional guidance embedding with the summed time–label embedding gives a 1,184-dimensional input to the coefficient MLP, whose architecture is

$$\mathbb{R}^{1184} \xrightarrow{\text{Linear}(1184,256)} \xrightarrow{\text{SiLU}} \xrightarrow{\text{Linear}(256,256)} \xrightarrow{\text{SiLU}} \xrightarrow{\text{Linear}(256,2)} \mathbb{R}^2.$$

The two outputs parameterize A_ϕ and B_ϕ. All linear layers include biases. The complete head, including the guidance embedder, contains 371,938 trainable parameters, approximately 0.055% of the 675 million parameters in the frozen SiT-XL vector field.

For conditional head training, we first construct an offline dataset of 5 million samples from randomly sampled (image, noise, time) tuples, with $t \sim \text{Unif}(0, 1)$, retaining each image's class label s. At each interpolated state and time, we evaluate the conditional velocity using s and the unconditional velocity using the null condition. Following Appendix A.6, we compute the four branch-level JVPs with respect to state and time, holding the conditioning inputs fixed, and store the branch velocities and JVP outputs needed for coefficient fitting. Thus, label information enters both the coefficient head and the conditional branch used to construct its acceleration targets. We then train the head for 15 epochs over this dataset, randomly sampling guidance scales from $\{0, 1, 1.375, 1.5\}$ and reconstructing the guided velocities and accelerations from the stored branch quantities without additional base-model or JVP evaluations.

Figure 5 shows the acceleration and learned-head residual profiles for unconditional, conditional, and CFG generation. At each evaluated time t, we estimate both profiles using the same 5,000 forward data-noise pairs. For the conditional and CFG profiles, we randomly select five data samples from each of the 1,000 classes and pair them with noise, giving equal representation to every class.

Across all four settings in Figure 5, the learned head consistently reduces the estimated expected residual norm relative to the Euler residual. The reduction is particularly pronounced near the data endpoint, where the acceleration norm rises sharply. The predominantly positive median paired reductions indicate that the gains are not driven solely by a small number of samples, although individual samples can exhibit increased residuals. These results support the state-velocity span as a useful inductive bias for approximating the learned field's acceleration, without implying that the acceleration lies exactly in this span. The profiles assess acceleration approximation on forward-interpolant samples. They do not by themselves establish improved generated-sample quality.

Table 5 reports the propagation parameters used for the ImageNet-256 experiments. For each sampler-schedule, guidance, and NFE setting, we select the value of L with the lowest FID in a 3,000-sample sweep. We use $L \in \{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ for $w = 1.375$ and $L \in \{1.0, 1.5, 2.0, 2.5, 3.0\}$ for $w \in \{1, 1.5\}$.

A.9.4 Sensitivity to the propagation parameter L

We assess sensitivity to L on ImageNet-256 using SAS-DPR at $w = 1.5$ and total NFE 32. We test

$$L \in \{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}.$$

Across these settings, we keep the pretrained velocity field, coefficient head, empirical residual profile, and evaluation noise samples fixed, varying only L when constructing the residual-aware schedule. Each configuration is evaluated using 50,000 generated samples.

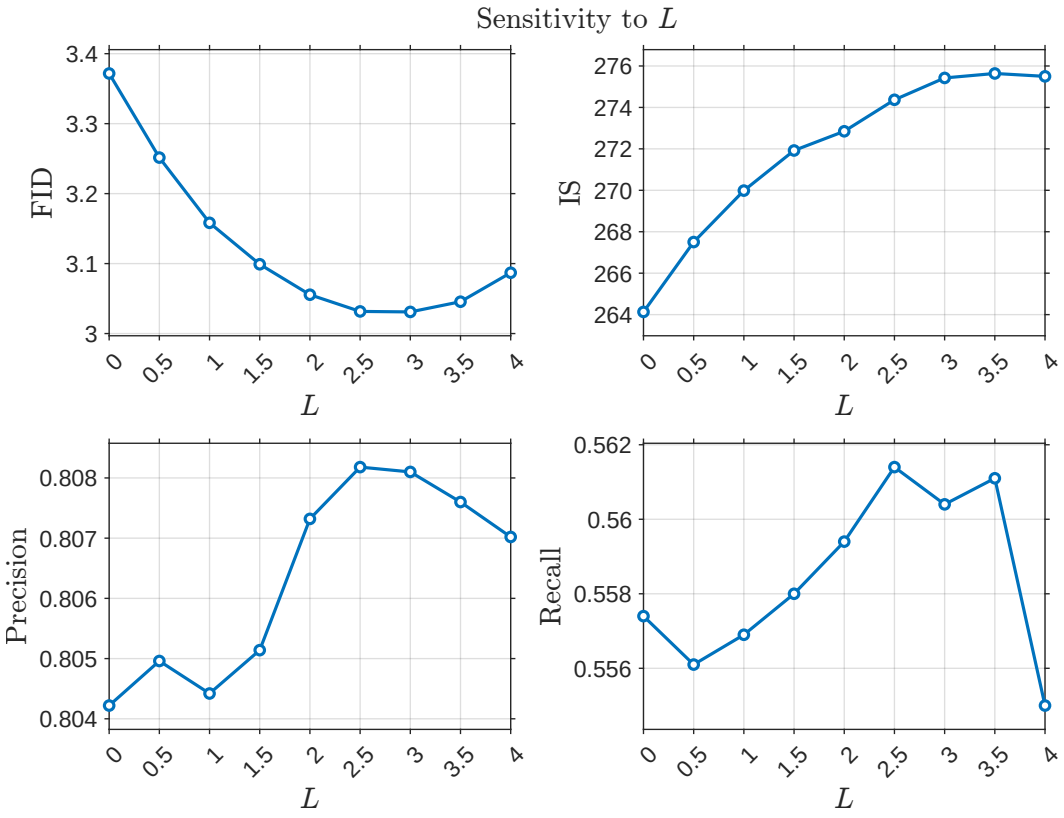

Figure 6: Sensitivity to the propagation parameter L on ImageNet-256 with SiT-XL, SAS-DPR, guidance scale $w = 1.5$, and total NFE 32. Lower FID and higher IS, precision, and recall are better.

Figure 6 shows that performance improves as L increases from zero to moderate values and then remains relatively stable. In particular, FID, IS, precision, and recall all vary modestly over $L \in [2, 3.5]$, indicating that the method does not require a narrowly tuned propagation parameter within this range. The SAS-DPR result in Table 2 uses $L = 2.0$, selected by the 3,000-sample sweep. Compared with the post hoc 50,000-sample sweep in Figure 6, its FID of 3.06 is close to the best FID of 3.03 at $L = 3.0$, while recall is unchanged at the displayed precision: 56.0% at $L = 2.0$ versus 56.0% at $L = 2.5$. The endpoint results provide some qualification: $L = 0$ gives worse FID and IS, while $L = 4$ slightly worsens FID and produces a more visible recall drop. Thus, the method is not very sensitive to L over a broad moderate range, although extreme values can degrade individual metrics.

A.10 Limitations and Computational Complexity

We discuss the limitations of the structured approximation and residual-based scheduling, clarify the scope of the theoretical and empirical results, and summarize the offline and sampling costs.

Approximation and sample quality. The state-velocity span is an inductive bias: two scalar coefficients can represent only acceleration within this span. Neither the forward acceleration identity nor its higher-derivative generalization implies that the corresponding reverse-flow derivatives lie in the same span, so a nonzero approximation residual can remain. Moreover, fitting the head approximates the frozen field's acceleration rather than eliminating model mismatch. A smaller computable residual therefore does not by itself guarantee improved FID or recall at every sampling budget.

Residual profiles and timestep allocation. For CFG at guidance scales other than 0 or 1, forward-interpolant samples serve as a proxy for the guided-state distribution. The resulting residual profiles

may therefore differ from those encountered during guided sampling. The timestep schedule is a fixed global grid rather than an instance-adaptive rule, and its construction depends on the empirical profile, candidate grid, and propagation parameter L. In practice, L is tuned rather than obtained as a certified Lipschitz constant.

Theoretical and empirical scope. Theorem 2 applies to the structured one-evaluation update under the stated regularity and coefficient-boundedness assumptions. Transferring the Euler-residual schedule to Heun, AB2, RF-Solver, or other integrators is empirical and does not by itself establish corresponding error or convergence guarantees. The experimental evidence is limited to the tested image-generation models, guidance settings, solver families, and evaluation budgets; gains need not extend uniformly beyond these settings.

Offline cost and computational complexity. The extra cost is primarily offline and includes JVP target collection, storage, head training, residual profiling, and schedule construction. Computations were performed primarily on an NVIDIA RTX 4090, with three RTX A6000 GPUs used for a subset of the workload. Because GPU availability and process concurrency varied, the timings below are approximate GPU-hours rather than measurements obtained with a fixed GPU count or fixed wall-clock configuration.

For CFG, (37) uses four branch-level JVPs per derivative record, reusable across guidance scales. For ImageNet-256 with SiT-XL, the coefficient-head training dataset contains 5 million conditional records and therefore requires 20 million branch-level JVP evaluations. Collecting these records required approximately 185 GPU-hours, primarily on the RTX 4090. The stored dataset occupies approximately 230 GB. The $C_0 = Q_{\mathrm{u}0}$, $C_1 = Q_{\mathrm{c}0} - Q_{\mathrm{u}0} + Q_{\mathrm{u}1}$, $C_2 = Q_{\mathrm{c}1} - Q_{\mathrm{u}1}$, z, v_θ^{c}, andv_θ^{u} tensors are stored in `bfloat16`; the time tensor is stored in `float32`, and the class-label tensor is stored in 64-bit integer. Once these records are available, training the lightweight coefficient head for 15 epochs takes around 4 hours on a single RTX 4090. Thus, JVP collection, rather than head fitting, dominates this stage of the offline cost.

Residual profiling is performed separately from training-data collection. For SiT-XL, we use 5,000 images, comprising five images per ImageNet class, and evaluate each image at 1,000 uniformly spaced times. Four branch-level JVPs are computed for every image-time pair, giving an additional 20 million JVP evaluations. This profiling required approximately 160 GPU-hours, again primarily on the RTX 4090. These one-time offline costs are amortized across guidance scales, schedule construction, and subsequent sampling runs.

At sampling time, structured acceleration adds only the lightweight coefficient-head evaluation and no additional base-model evaluation. The head adds only 0.055% as many parameters as the frozen SiT-XL vector field, and its sampling-time overhead is negligible in our implementation. For the candidate grids used in our experiments, schedule construction takes only a few seconds and is negligible relative to JVP collection and residual profiling. For a candidate grid with K intervals and N sampling steps, interval-cost preprocessing takes $\Theta(K)$ time using prefix sums, while the dynamic program takes $\Theta(NK^2)$ time and $\Theta(NK)$ memory for its values and backtracking pointers.