THE DENOISING ESTIMATE IS A STATE VARIABLE: JOINT REFINEMENT FOR FEW-STEP FLOW LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow language models (FLMs) generate text by repeatedly predicting a clean representation with a denoiser and using it to transport a noisy latent state toward the data distribution. The first-order samplers commonly used in FLMs evaluate the denoiser once at the current state and keep the resulting estimate fixed throughout the update. This approximation becomes inaccurate for large steps, because changing the estimate also changes the transported trajectory and hence the state at which subsequent predictions are evaluated. We introduce **AIM-FLM**, which refines the estimate jointly with the state it induces. Given a candidate estimate, AIM-FLM constructs the induced intermediate state through the closed-form affine transport of the flow, re-evaluates the denoiser at that state, and repeats this Picard iteration toward a joint fixed point. Under linear interpolation schedules this fixed point corresponds to an implicit-midpoint step, but it is solved directly in the denoising-estimate space, preserving a clean representation that can be decoded at the endpoint and retained as feedback across solver steps or distilled hops. Under a local contraction condition, the iteration converges geometrically, and under a margin condition, the decoded tokens stabilize after finitely many iterations without requiring exact convergence of the continuous estimate. The same coupling applies to pretrained deterministic and stochastic samplers without retraining, can be preserved in flow-map distillation by passing the estimate from one step to the next, and can be incorporated into joint training of the denoising field and hop model. Across several flow language model backbones, AIM-FLM improves the trade-off between generation quality and compute. Without retraining, AIM-FLM surpasses the 32-NFE ELF-B sampler in generative perplexity using 17 denoiser evaluations, with comparable entropy and lower repetition. With joint training, it surpasses the same baseline using only 12.

1 INTRODUCTION

Continuous diffusion language models have emerged as a promising alternative to autoregressive generation by modeling language in continuous representation spaces Li et al. (2022); Gong et al. (2022); Lovelace et al. (2023). Within this family, flow language models (FLMs) Lipman et al. (2022); Liu et al. (2022b); Hu et al. (2026) generate text by integrating a probability-flow ODE (PF-ODE) Song et al. (2020) and can in principle use far fewer steps than discrete diffusion language models Zheng et al. (2025); Lee et al. (2026). At each solver step, the denoiser predicts a clean representation, and this estimate plays two roles. It predicts the clean data, and it parameterizes the affine transport that determines the next latent state, which we call its induced state. Existing refinements improve the estimate while keeping this state fixed. Self-

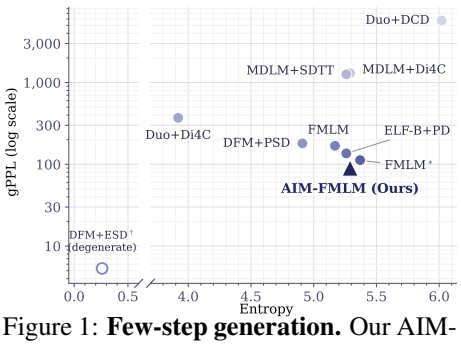


Figure 1: **Few-step generation.** Our AIM-FLM achieves low generative perplexity while maintaining generation diversity.

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.

conditioning Chen et al. (2022) feeds a previous estimate back as an auxiliary input, fixed-state refinement Yoo et al. (2026) iterates this feedback toward a fixed point at the current latent state, and the first-order samplers commonly used in FLMs Hu et al. (2026); Chen et al. (2026) hold the estimate fixed across each update. This approximation is accurate for small steps, since the latent state barely moves within an update Liu et al. (2022a). In few-step regimes such as large solver steps and flow-map distillation, however, a single estimate drives the state over a long interval Salimans & Ho (2022), and an estimate refined at a fixed state need not agree with the denoiser evaluated at its own induced state. Standard explicit higher-order solvers Lu et al. (2022); Karras et al. (2022) evaluate the denoiser at intermediate states constructed from preceding stage estimates, but do not solve the estimate-level fixed point considered here.

We propose **AIM-FLM**, which refines the clean estimate jointly with the state it induces. Given a candidate estimate, AIM-FLM uses the closed-form affine PF-ODE transport to construct its induced midpoint state, evaluates the denoiser there, and takes the output as the next candidate. Repeating these steps drives the estimate toward a joint fixed point at which it reproduces itself when the denoiser is evaluated at its own induced midpoint. Under linear interpolation schedules this fixed point is an implicit-midpoint step of the PF-ODE, and one iteration from the first-order estimate recovers the explicit midpoint update. Because the fixed point is solved in the space of the clean estimate, the converged variable can be decoded into tokens directly and carried into later steps. The coupling needs no self-conditioning channel, since the estimate already reaches the denoiser through its induced state, although such a channel can still be used when available. In practice we run a few Picard iterations Atkinson (2008). We prove local geometric convergence under a contraction condition and bound the error after a finite budget. Once this error falls below the decoding margin, the decoded tokens coincide with those of the exact fixed point, even before the continuous estimate converges.

The same issue is amplified in flow-map distillation Boffi et al. (2024; 2026), where one student prediction replaces several teacher steps. Representative flow-map students Yoo et al. (2026); Lee et al. (2026) primarily supervise the endpoint latent state of each hop, without explicitly preserving the outgoing denoising estimate as a separate transition variable. Exploiting the affine structure of the teacher trajectory, AIM-FLM trains the student to predict the network-dependent part of the endpoint together with the outgoing estimate. This preserves the feedback between hops and permits hop-level refinement at inference. The denoising field and the hop model can also be trained jointly under this formulation, so that the coupling is learned from scratch rather than added post hoc. We evaluate AIM-FLM in training-free sampling, flow-map distillation, and joint training, counting refinement iterations in all NFEs. Without retraining, AIM-FLM surpasses the 32-NFE ELF-B Hu et al. (2026) sampler in generative perplexity using 17 NFEs, with comparable entropy and lower repetition. In few-step distillation, it improves generative perplexity under one-, two-, and four-step budgets while keeping generation entropy non-degenerate (see Fig. 1). With joint training, it outperforms the 18-NFE ELF-B baseline with only 3 NFEs and surpasses the 32-NFE baseline with 12. Our contributions are summarized as follows.

- We show that in FLMs the clean estimate both predicts the clean data and determines the next latent state, and that refining it at a fixed state does not ensure consistency with its induced trajectory in few-step regimes.
- We introduce AIM-FLM, a joint refinement method built on the closed-form affine PF-ODE transport, which corresponds to an implicit-midpoint step under linear schedules and applies to pre-trained deterministic and stochastic samplers without retraining. We establish local geometric convergence of finite Picard refinement and a finite-iteration condition for stable token decoding.
- We extend the formulation to flow-map distillation by supervising the outgoing estimate alongside the endpoint, and to joint training of the denoising field and hop model.

2 PRELIMINARIES

Flow language models. Let $\mathcal{E}$ map a token sequence y to a continuous representation $x = \mathcal{E}(y)$, Flow language models (FLMs) define noisy latent states as $z_\rho = \alpha(\rho)x + \sigma(\rho)\varepsilon$, where $\varepsilon \sim \mathcal{N}(0, I)$ and ρ denotes the native noise-to-data coordinate and $\alpha(\rho), \sigma(\rho)$ specify the interpolation schedule. For $\sigma(\rho) > 0$, the corresponding velocity field admits the x-prediction form, $v(z_\rho, \rho) = \lambda(\rho)x + \tau(\rho)z_\rho$, where $\lambda(\rho)$ and $\tau(\rho)$ are two predefined coefficients. We therefore write $D_\theta(z_\rho, \rho; c)$ for

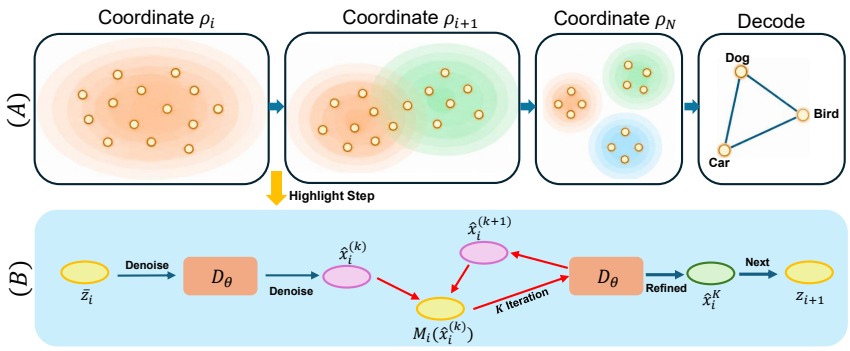


Figure 2: **Illustration of the joint implicit-midpoint refinement sampler. (A)** shows the standard PF-ODE sampling process discussed in Sec. 2. **(B)** illustrates the refinement mechanism within a single coordinate step, where the estimate and the induced latent state are iteratively updated to improve step-level consistency. The red arrow highlights our joint refinement method (i.e., Sec. 3.1).

the denoiser that predicts the clean representation x, where c denotes an optional self-conditioning input. Sampling follows the associated PF-ODE,

$$\frac{dz_\rho}{d\rho} = \lambda(\rho) D_\theta(z_\rho, \rho; c) + \tau(\rho) z_\rho.$$

A key property used throughout this work is that, when a candidate clean estimate $\hat{x}$ is held fixed over an interval, the state admits the closed-form affine transport,

$$\mathcal{P}_{\rho \to \rho'}(z; \hat{x}) = \frac{\sigma(\rho')}{\sigma(\rho)} z + \left(\alpha(\rho') - \frac{\sigma(\rho')}{\sigma(\rho)} \alpha(\rho) \right) \hat{x}, \qquad \sigma(\rho) > 0. \tag{1}$$

Thus, the denoising estimate both predicts clean content and determines the state reached by the solver. Importantly, Eq. 1 is exact for a held-fixed estimate. The approximation arises from freezing the estimate over the interval, not from the state transport itself. **See Appendix A.1 for more details**. Some FLM samplers additionally re-noise the latent state after each transport step. We write this re-noising step in the general form,

$$\bar{z}_i = \kappa_i z_i + \eta_i \xi_i, \qquad \xi_i \sim \mathcal{N}(0, I), \tag{2}$$

where κ_i and η_i are determined by the sampling protocol. The re-noised state $\bar{z}_i$ is associated with coordinate $\bar{\rho}_i$, after which Eq. 1 transports it from $\bar{\rho}_i$ onward. Deterministic sampling is recovered with $\kappa_i = 1$, $\eta_i = 0$, and $\bar{\rho}_i = \rho_i$. The specific coordinates and re-noising rule used by each FLM family are given in **Appendix B**.

Self-conditioning and fixed-state refinement. Self-conditioning feeds the denoiser an earlier clean estimate as the auxiliary input c. During training, this is typically a same-coordinate surrogate, $c = \mathrm{sg}[D_\theta(z_\rho, \rho; \varnothing)]$, where $\varnothing$ denotes null conditioning and sg is the stop-gradient operator. At inference, samplers commonly reuse an estimate produced at a preceding sampling point, creating a mismatch between the state and noise level at which the estimate is produced and those at which it is reused. Additionally, at a pinned state (z_ρ, ρ), the conditioning channel defines the self-map, $\hat{x}^{(k+1)} = D_\theta\big(z_\rho, \rho; \hat{x}^{(k)}\big)$, whose fixed point satisfies,

$$\hat{x}^\star = D_\theta\big(z_\rho, \rho; \hat{x}^\star\big). \tag{3}$$

This makes the estimate consistent with its own conditioning input at the pinned state. However, Eq. 1 shows that changing $\hat{x}$ also changes the state transported by the solver. Therefore, estimate consistency at a fixed z_ρ does not imply consistency with the trajectory induced by that estimate. This state–estimate coupling motivates the joint refinement introduced next.

3 Methodology

3.1 Joint implicit-midpoint refinement

We seek consistency between the denoising estimate and the state induced by that estimate. Consider step i from the re-noised state $\bar{z}_i$ at $\bar{\rho}_i$ to z_{i+1} at ρ_{i+1}, and define the midpoint coordinate $\rho_{m,i} =$

$\frac{1}{2}(\bar{\rho}_i + \rho_{i+1})$. For a candidate estimate $\hat{x}$, we use Eq. 1 to define the induced midpoint state and the corresponding denoiser update,

$$M_i(\hat{x}) = \mathcal{P}_{\bar{\rho}_i \to \rho_{m,i}}\big(\bar{z}_i; \hat{x}\big), \qquad \mathcal{F}_i(\hat{x}) = D_\theta\big(M_i(\hat{x}), \rho_{m,i}; c = \hat{x}\big), \tag{4}$$

where c is an optional self-conditioning input discussed below. A joint fixed point is a pair $(\hat{x}_i^\star, z_{m,i}^\star)$ satisfying $z_{m,i}^\star = M_i(\hat{x}_i^\star)$ and $\hat{x}_i^\star = \mathcal{F}_i(\hat{x}_i^\star)$. The corresponding solver step is then $z_{i+1}^\star = \mathcal{P}_{\bar{\rho}_i \to \rho_{i+1}}\big(\bar{z}_i; \hat{x}_i^\star\big)$. Unlike the fixed-state refinement in Eq. 3, the state at which the denoiser is evaluated now changes with the estimate being refined. The estimate is therefore required to be consistent with the trajectory it induces.

Implicit-midpoint interpretation. Assume the linear interpolant $\alpha(\rho) = \rho$ and $\sigma(\rho) = 1 - \rho$. Then, for any fixed estimate, the transport path of Eq. 1 is affine in ρ, so the induced midpoint state is the average of the start state and the endpoint transported with the same estimate. In particular, at a converged fixed point, $M_i(\hat{x}_i^\star) = (\bar{z}_i + z_{i+1}^\star)/2$. Let $h_i = \rho_{i+1} - \bar{\rho}_i$, and the solver step at the fixed point can then be written as $z_{i+1}^\star = \bar{z}_i + h_i\, v_\theta^\star\big(M_i(\hat{x}_i^\star), \rho_{m,i}\big)$, where $v_\theta^\star(z, \rho)$ denotes the velocity induced by the fixed-point denoising estimate. Intuitively, converged joint refinement moves the latent state along the direction predicted at the midpoint of the step, whereas fixed-state refinement uses the direction predicted at its start throughout the whole interval. The full derivation and analyses are provided in **Appendix A.3**.

Practical refinement. As shown in Eq. 4, this construction does not require a self-conditioning channel. When the backbone supports self-conditioning, we set $c = \hat{x}$ in $\mathcal{F}_i$, so that at convergence the conditioning input equals the estimate predicted at the induced midpoint state. Otherwise, we use $\mathcal{F}_i(\hat{x}) = D_\theta\big(M_i(\hat{x}), \rho_{m,i}\big)$. The iteration remains nontrivial because changing $\hat{x}$ still changes the state at which the denoiser is evaluated. In practice, starting from an initial estimate $\hat{x}_i^{(0)}$, we apply K Picard iterations $\hat{x}_i^{(k+1)} = \mathcal{F}_i(\hat{x}_i^{(k)})$ and complete the step with $z_{i+1} = \mathcal{P}_{\bar{\rho}_i \to \rho_{i+1}}(\bar{z}_i; \hat{x}_i^{(K)})$. Each Picard iteration requires one denoiser evaluation, making K a direct inference-time compute parameter.

3.2 Theoretical properties

We first characterize the finite-budget refinement. Since the transport in Eq. 1 is affine in the estimate, the midpoint state can be written as $M_i(\hat{x}) = a_i^{\mathrm{m}} \bar{z}_i + b_i^{\mathrm{m}} \hat{x}$. Then, we have

Proposition 3.1 (Local convergence of joint refinement)**.** *Fix the starting state $\bar{z}_i$, the noise draw ξ_i, and all sampler settings of step i. Let*

$$\mathcal{U}_i = \Big\{\hat{x} : \|\hat{x} - \hat{x}_i^{(0)}\| \le r_i\Big\}$$

be a ball on which D_θ is continuously differentiable, and define

$$q_i = \sup_{\hat{x} \in \mathcal{U}_i} \|b_i^{\mathrm{m}} \partial_z D_\theta + \partial_c D_\theta\|_{\mathrm{op}}, \qquad \delta_i = \Big\|\mathcal{F}_i(\hat{x}_i^{(0)}) - \hat{x}_i^{(0)}\Big\|. \tag{5}$$

The Jacobians are evaluated at $(M_i(\hat{x}), \rho_{m,i}; \hat{x})$, with $\partial_c D_\theta \equiv 0$ for backbones without self-conditioning. If

$$q_i < 1, \qquad \delta_i \le (1 - q_i) r_i,$$

then $\mathcal{F}_i$ has a unique fixed point $\hat{x}_i^\star \in \mathcal{U}_i$, the Picard iterates remain in $\mathcal{U}_i$, and for every $K \ge 1$,

$$\Big\|\hat{x}_i^{(K)} - \hat{x}_i^\star\Big\| \le \frac{q_i^K}{1 - q_i}\delta_i, \qquad \Big\|z_{i+1}^{(K)} - z_{i+1}^\star\Big\| \le |s_i|\frac{q_i^K}{1 - q_i}\delta_i, \tag{6}$$

where

$$s_i = \alpha(\rho_{i+1}) - \frac{\sigma(\rho_{i+1})\alpha(\bar{\rho}_i)}{\sigma(\bar{\rho}_i)}$$

is the estimate coefficient of the full-step affine transport.

Proposition 3.1 shows that the finite-iteration error decreases geometrically with K under the local contraction condition. The bound also depends on the initialization residual δ_i, motivating the warm-

start strategy used in Mode A. The two terms in Eq. 5 correspond to the state and conditioning pathways. Since the state-dependent term remains when $\partial_c D_\theta = 0$, the refinement does not rely on self-conditioning.

The continuous estimate, however, need not converge exactly for the decoded language output to stabilize. The following result connects the refinement error to the discrete token prediction.

Corollary 3.2 (Finite-step token stabilization)**.** *Let i denote the final refined step, at which the terminal readout is applied, and adopt the conditions of Proposition 3.1. Let $\mathcal{R}(\hat{x})$ denote the terminal token scores obtained from an estimate through the base model's endpoint reconstruction and readout, with any decoding randomness fixed. Suppose $\mathcal{R}$ is Lipschitz on $\mathcal{U}_i$ with constant L_K:*

$$\|\mathcal{R}(\hat{x}) - \mathcal{R}(\hat{x}')\|_\infty \le L_K \|\hat{x} - \hat{x}'\|_2, \qquad \forall\, \hat{x}, \hat{x}' \in \mathcal{U}_i.$$

At position ℓ, let $g_\ell > 0$ denote the gap between the largest and second-largest scores in $\mathcal{R}(\hat{x}_i^\star)$. Then the argmax prediction from $\hat{x}_i^{(K)}$ agrees with the fixed-point prediction whenever

$$\frac{2L_K q_i^K}{1 - q_i}\delta_i < g_\ell. \tag{7}$$

For a finite sequence with positive margins at every position, the decoded sequence therefore stabilizes after finitely many refinement steps.

Corollary 3.2 shows that exact convergence in the continuous representation space is not required for stable language decoding. Once the refinement error falls below the decoder margin, further iterations no longer change the argmax output. Detailed proofs of Proposition 3.1 and Corollary 3.2 are provided in **Appendix A.2**.

3.3 Joint refinement as a plug-and-play module

The same state–estimate structure can be used without training, preserved during flow-map distillation, or learned jointly with the denoising field. These settings differ in how many solver steps are represented by one model evaluation, while retaining the estimate that determines subsequent state transport.

Mode A: training-free sampling. Joint refinement can be applied to pretrained FLMs without modifying its architecture or parameters. At step i, we initialize the estimate using the previous two estimates,

$$\hat{x}_i^{(0)} = \hat{x}_{i-1} + \omega\big(\hat{x}_{i-1} - \hat{x}_{i-2}\big).$$

When only one previous estimate is available, we use $\hat{x}_{i-1}$, and at the first step we follow the initialization of the base sampler. We then apply Eq. 4 for a finite number of Picard iterations and complete the step using Eq. 1. In the default configuration, we use one Picard iteration per coordinate. The warm start is used to reduce the initial residual δ_i in Eq. 6. The conditioning input is omitted when it is not supported by the backbone. Backbone-specific sampling details are provided in **Appendix B.1**.

Mode B: feedback-preserving hop distillation. We next compress a segment of refined teacher steps into a single learned hop from z_{T_j} to $z_{T_{j+1}}$. Once the refined estimate $\hat{x}_i$ and noise draw ξ_i are fixed, Eq. 1 and Eq. 2 imply that each teacher step can be written as

$$z_{i+1} = a_i z_i + b_i \xi_i + v_i \hat{x}_i,$$

where the coefficients are fixed by the coordinate grid and re-noising schedule. Composing these transitions over the teacher segment $[T_j, T_{j+1})$ gives

$$z_{T_{j+1}} = A_j z_{T_j} + \underbrace{\sum_{i=T_j}^{T_{j+1}-1} B_{ji}\xi_i}_{N_j} + \underbrace{\sum_{i=T_j}^{T_{j+1}-1} V_{ji}\hat{x}_i}_{G_j}, \tag{8}$$

where

$$A_j = \prod_i a_i, \qquad B_{ji} = b_i \prod_{l>i} a_l, \qquad V_{ji} = v_i \prod_{l>i} a_l.$$

The terms $A_j z_{T_j}$ and N_j are determined by the incoming state, schedule, and sampled noise. The network-dependent contribution to the endpoint is therefore summarized by the content aggregate G_j.

A one-hop student does not observe the intermediate estimates needed to construct G_j. We form a surrogate by holding the incoming estimate $\hat{x}_{T_j}$ fixed across the segment,

$$\tilde{G}_j = V_j \hat{x}_{T_j}, \qquad V_j = \sum_i V_{ji},$$

which gives

$$\tilde{z}_{T_{j+1}} = A_j z_{T_j} + N_j + \tilde{G}_j.$$

The student H_ϕ acts as a corrector of this endpoint guess. It predicts both the content aggregate and the outgoing estimate:

$$\big(\hat{G}_j, \hat{x}_{T_{j+1}}\big) = H_\phi\big(\tilde{z}_{T_{j+1}}, \rho_{T_{j+1}}; c = \hat{x}_{T_j}\big), \qquad \hat{z}_{T_{j+1}} = A_j z_{T_j} + N_j + \hat{G}_j. \tag{9}$$

The two outputs play different roles. The aggregate $\hat{G}_j$ determines the transported endpoint state, while $\hat{x}_{T_{j+1}}$ is carried to the next hop and determines subsequent transport. The student therefore preserves both components of the state–estimate dynamics rather than distilling only the latent endpoint. Training minimizes the difference between $\hat{z}_{T_{j+1}}$ and the teacher endpoint $z_{T_{j+1}}$, together with a regularization term on the outgoing estimate $\hat{x}_{T_{j+1}}$. Implementation and training details are provided in **Appendix B.2**.

Preserving the outgoing estimate also allows the hop itself to be refined at inference time. Given an updated estimate, we recompute $\tilde{G}_j$ and the student input, evaluate H_ϕ again, and feed the returned estimate into the next iteration. This defines a hop-level fixed-point iteration analogous to Eq. 4. The number of hop evaluations can therefore be varied at inference time without retraining. Its convergence analysis is provided in **Appendix A.5**.

Mode C: joint field and hop learning. Finally, the denoising field and hop model can be trained together. We optimize the backbone D_θ with its original FLMs objective, while H_ϕ follows the Mode B distillation objective using trajectories from an exponential moving average of the backbone. Each feedback pass constructs its input from the predicted aggregate $\hat{G}_j$, and all passes are supervised against the same target. To support different hop lengths, we additionally train hops spanning two adjacent segments. Their target is the average of two estimates constructed from the same context: one obtained by running the EMA field over the full span and one obtained by composing two shorter EMA hops. The first anchors the long hop to the underlying field trajectory, while the second keeps it consistent with the short-hop composition. At inference, the hop model uses the same feedback mechanism as Mode B. Full training details are provided in **Appendix B.3**.

4 Experiments

We evaluate three consequences of the state-variable view developed in Sec. 3. $\boldsymbol{Q}_1$ Does coupling the denoising estimate with its induced latent state improve inference-time sampling beyond fixed-state refinement under a matched compute budget? $\boldsymbol{Q}_2$ Does preserving state–estimate feedback improve few-step flow-map distillation, and can the resulting hop student continue to benefit from inference-time refinement? $\boldsymbol{Q}_3$ Can the same structure be learned jointly with the denoising field rather than introduced through post-hoc distillation?

General Setup. We evaluate on OpenWebText Gokaslan & Cohen (2019) with a fixed generation length of 1,024 tokens. We report generative perplexity (gPPL) under GPT-2 Large and mean per-sample empirical unigram entropy in nats. For training-free sampling, we evaluate released FLMs backbones from the ELF families Hu et al. (2026), LangFlow Chen et al. (2026), and FLM Lee et al. (2026). For few-step generation, we additionally compare with Duo using DCD and Di4C Sahoo et al. (2025), MDLM using SDTT and Di4C Chemseddine et al. (2026), DFM Potaptchik et al.

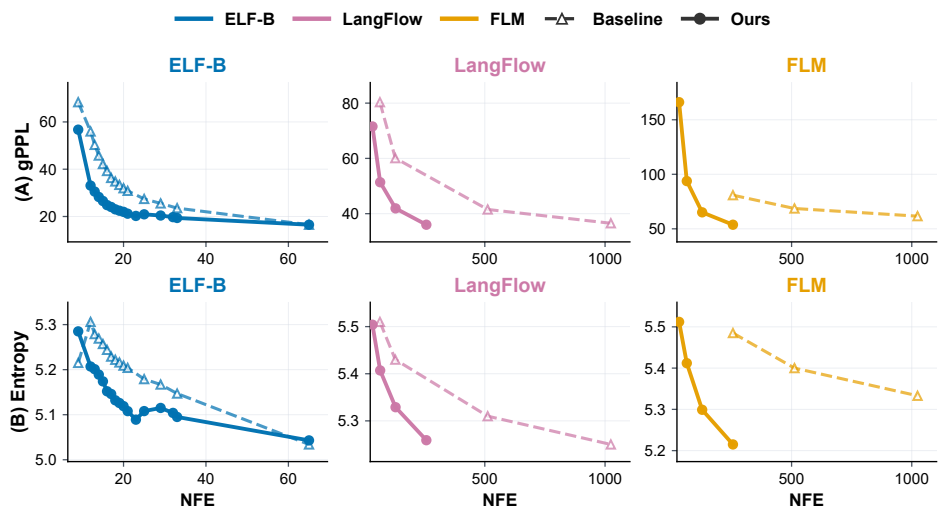


Figure 3: **Training-free joint implicit-midpoint refinement.** **(A)** Generative perplexity and **(B)** generation entropy across NFE budgets and FLMs backbones. AIM-FLM reaches lower gPPL at a given compute budget while maintaining generation diversity.

(2026), FMLM Lee et al. (2026), and FMLM* Yoo et al. (2026). We follow the corresponding baseline inference settings, including their noise schedules and sampling configurations. Unless stated otherwise, Mode B uses ELF-B as the teacher and one Picard iteration to construct the teacher trajectory. Implementation details for Modes A, B, and C are provided in **Appendix B.1**, **Appendix B.2**, and **Appendix B.3**, respectively. Additional experimental results and controls are provided in **Appendix C**.

A_1. Coupling the estimate with its induced latent state improves inference-time sampling beyond fixed-state refinement. We first apply joint implicit-midpoint refinement to pretrained FLMs without changing their parameters or training procedure. As shown in Fig. 3, AIM-FLM reduces gPPL at a given NFE budget across the evaluated backbones while keeping generation entropy stable. In the LangFlow setting, for example, AIM-FLM reaches a comparable quality regime with roughly 128 model evaluations, whereas the original sampler requires roughly 512. This result shows that the proposed refinement can improve the quality–compute trade-off of an existing FLM without retraining the backbone. We next isolate the source of this gain. In particular, an improvement from Picard refinement alone would not establish that the state–estimate coupling in Sec. 3.1 is necessary. We therefore fix the LangFlow backbone and coordinate schedule and compare two refinement procedures under the same inner iteration budget. The fixed-state variant updates the estimate while holding the latent state fixed, whereas AIM-FLM recomputes the induced state before the next denoiser evaluation. As shown in Fig. 4, both variants improve as the refinement budget increases, but joint refinement consistently reaches lower gPPL. The gap is about 120 after one inner iteration and remains about 60 at $k = 4$. Increasing the number of outer sampling steps improves both variants, while a gap of roughly 30 remains. Because the backbone, schedule, and refinement budget are matched, this comparison isolates the contribution of updating the state together with the estimate. This result directly supports the central claim of AIM-FLM, namely that consistency at a pinned state does not imply consistency with the trajectory induced by the estimate. Additional

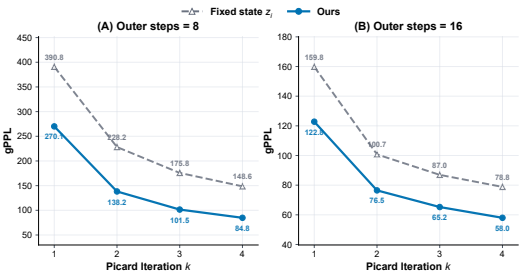


Figure 4: **Joint versus fixed-state refinement.** With the backbone and coordinate schedule fixed, joint refinement achieves lower gPPL than fixed-state refinement across Picard and outer-step budgets.

Table 1: **Few-step generation on OpenWebText.** AIM-FLM achieves the lowest gPPL among non-degenerate methods across one-, two-, and four-step budgets while maintaining comparable generation entropy. Best results are shown in **bold** and second-best non-degenerate results are underlined. † denotes a degenerate low-entropy solution.

		1 step		2 steps		4 steps	
Family	**Method**	**gPPL↓**	**Ent.↑**	**gPPL↓**	**Ent.↑**	**gPPL↓**	**Ent.↑**
ELF-B	PD	136.10	5.26	68.25	5.24	34.33	5.16
Duo	DCD	5743.29	6.02	891.16	5.41	250.86	5.37
	Di4C	370.51	3.92	210.22	4.63	154.67	4.85
MDLM	SDTT	1260.86	5.26	877.22	5.34	339.73	5.38
	Di4C	1298.80	5.29	758.23	5.35	239.27	5.40
FMLM		168.30	5.17	133.29	5.25	111.31	5.26
FMLM*		112.52	5.37	94.74	5.45	75.22	5.41
DFM	PSD	180.29	4.91	152.83	5.03	122.32	5.10
	ESD	5.33†	0.26†	108.91	5.15	77.08	5.27
Ours		**89.81**	5.29	**66.35**	5.24	**26.41**	5.12

cross-backbone and sampler results in **Appendix C.1** show that the gain extends across model families, including a backbone without self-conditioning.

A_2. Feedback-preserving hop distillation improves few-step generation and retains inference-time refinement. We next test whether the same state–estimate structure remains useful after multiple teacher steps are compressed into a learned hop. Following Sec. 3.3, the student predicts the network-dependent content aggregate together with the outgoing denoising estimate. The latter is carried into the next hop because it determines subsequent transport rather than serving only as an auxiliary conditioning input. Tab. 1 compares AIM-FLM after distillation (i.e., AIM-FMLM) with existing few-step generation methods at one-, two-, and four-step budgets. AIM-FLM obtains gPPL values of 89.81, 66.35, and 26.41, respectively. Relative to FMLM*, which performs fixed-state refinement, these correspond to approximately 20%, 30%, and 65% lower gPPL across the three budgets. The improvements are obtained while maintaining non-degenerate generation entropy. In particular, DFM-ESD reaches a raw gPPL of 5.33 in the one-step setting, but its entropy drops to 0.26, indicating a collapsed solution; AIM-FLM instead obtains 89.81 gPPL with entropy 5.29. The comparison therefore shows that the gain in gPPL does not rely on the low-diversity failure mode observed in the degenerate baseline. These results use teacher trajectories constructed with only one Picard iteration. We further test whether preserving the outgoing estimate allows the distilled student itself to remain refinable at inference time. Using LangFlow as the teacher backbone, we train H_ϕ under two hop settings and vary the number of Picard iterations used by the student. As shown in Fig. 5, additional refinement consistently reduces student perplexity. With sufficient refinement, the student can reach or surpass the teacher trajectory used for distillation. This behavior is enabled by preserving the estimate that determines the next transported state. As a result, the distilled hop model retains an inference time compute–quality trade-off rather than collapsing the teacher trajectory into a fixed one-shot state transition. **Appendix C.2** further shows that the same formulation compresses longer teacher trajectories across multiple FLMs families and sampler settings.

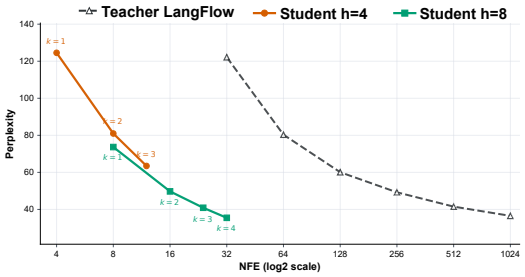


Figure 5: **Distilled students remain refinable at inference time.** Orange and green curves show students with $h = 4$ and $h = 8$ hops. k denotes the number of Picard iterations per coordinate, so the student NFE is kh.

A_3. Joint field-and-hop training learns the same few-call structure from scratch. The previous results use a pretrained teacher trajectory. We finally test whether the state–

estimate structure can instead be learned jointly by training the hop module together with the base denoising field. Following Mode C in Sec. 3.3, we use the ELF-B architecture for the denoising field and construct hop targets from EMA trajectories during training. As shown in Tab. 2, joint training preserves the few-call behavior observed after post-hoc distillation. With only 3 NFEs, AIM-FLM obtains 34.00 gPPL, slightly improving over the 34.83 gPPL of the 18-NFE ELF-B sampler. With 6 NFEs, AIM-FLM reaches 23.84 gPPL, nearly matching the 23.54 gPPL obtained by ELF-B with 32 NFEs. Increasing the budget to 12 NFEs further reduces gPPL to 20.37 while maintaining similar generation entropy. **Appendix C.3** provides matched-field controls and shows that concurrent hop training remains effective beyond post-hoc training.

Sampler	h	k	**NFE**↓	**gPPL**↓	**Ent.**
ELF-B	–	–	18	34.83	5.22
AIM-FLM	3	1	**3**	**34.00**	5.08
ELF-B	–	–	32	**23.54**	5.15
AIM-FLM	3	2	**6**	23.84	5.08
ELF-B	–	–	64	**16.52**	5.03
AIM-FLM	6	2	**12**	20.37	5.07

Table 2: **Joint training enables few-call generation.** AIM-FLM reaches comparable ELF-B quality with substantially fewer neural function evaluations. h denotes hop size and k the number of Picard iterations.

Taken together, $\boldsymbol{A}_1$ isolates the effect of coupling the estimate with its induced state under matched refinement compute, $\boldsymbol{A}_2$ shows that this state–estimate feedback can be preserved when several solver steps are compressed into a few-step student, and $\boldsymbol{A}_3$ shows that the same structure can be learned jointly from scratch. The three settings therefore test the same underlying claim at the solver-step, distilled-hop, and jointly trained model levels.

5 Related Work

Continuous diffusion language models. Recent progress in diffusion models for continuous domains has motivated growing interest in adapting diffusion-style generation to language Jo & Hwang (2026); Lee et al. (2026); Gong et al. (2022); Han et al. (2023); Dong et al. (2026); Guo et al. (2026b). Since language is inherently discrete, a central challenge is to design continuous representations and denoising dynamics that support efficient generation while preserving sequence-level semantic structure Jo & Hwang (2026). Continuous diffusion language models address this challenge by learning denoising dynamics in embedding spaces through score matching Li et al. (2022); Dieleman et al. (2022); Lovelace et al. (2023) or flow matching. Among them, flow language models Hu et al. (2026); Chen et al. (2026); Lee et al. (2026); Yoo et al. (2026); Guo et al. (2026a)have attracted increasing attention because their probability-flow formulation provides smooth continuous trajectories and supports efficient few-step sampling. However, these works primarily focus on representation design, training objectives, or flow-map distillation, while leaving the local consistency of the sampler largely unexplored. Our work instead revisits the sampler operator itself and improves each step by coupling the denoising estimate with the latent state it transports, yielding a plug-in refinement mechanism for existing flow language models.

6 Conclusion

Motivated by the affine transport property of noisy latent states and clean estimates, we propose joint implicit midpoint refinement for more consistent step-level sampling in FLMs. Specifically, our method couples the denoising estimate with the latent state it induces through deterministic transport, and evaluates refinement at an estimate-dependent midpoint. We also provide the theoretical analysis to establish its local convergence under contraction conditions and show that finite refinement can be sufficient for consistent token decoding. As a result, the proposed refinement can be used as a plug-in module for existing FLM architectures, including inference without additional training, feedback preserving hop distillation, and joint field and hop learning. In particular, our hop distillation goes beyond direct state distillation by preserving the accumulated feedback from intermediate estimates, allowing the student to trade additional evaluations for higher generation quality. Based on this framework, we introduce AIM-FLM, an efficient flow map language model that improves generation quality over existing FMLMs baselines. Experiments further validate the effectiveness of our method both as a sampler and under joint training.

AI USE STATEMENT

We used generative AI tools to polish the writing of this paper, including grammar, wording, and readability. We did not use generative AI tools for research ideation, experimental design, running experiments, data analysis, theoretical derivations, or literature search. The authors reviewed every AI-assisted edit against the underlying results and take full responsibility for the content of the paper.

ETHICS STATEMENT

This work presents AIM-FLM, a methodological study aimed at improving the sampling efficiency of FLMs. It does not involve human subjects, crowdsourcing, or the collection of new personal data. Following the experimental protocols of prior FLM work, all models are trained and evaluated exclusively on publicly released datasets and benchmarks, such as OpenWebText. Pretrained models, including GPT-2 Large, are used solely as fixed evaluators to compute automatic metrics such as generative perplexity, and no human annotators are involved at any stage of evaluation. As with other text generation models, few-step FLMs could in principle be misused to produce misleading or harmful content at scale; however, our contribution concerns sampling efficiency and does not expand model capabilities beyond those of existing publicly available language models.

REPRODUCIBILITY STATEMENT

Modes A and B operate on the publicly released checkpoints of ELF-B/M/L, LangFlow and the one-hot FMLM base, with no change to their architectures, weights or decoders; the trained parameters of Mode B are the students described in Appendix B.2, and Mode C trains its own ELF-B field with the released ELF recipe together with the hop parameters of Appendix B.3. Appendix B.1 specifies the Mode A sampler completely, including the frozen grid and its seed, the re-noising rule, the initialization and refinement of each step, the three readouts and the NFE accounting, and the configuration and reference setting of every row of Table 3. Appendix B.2 specifies, for every student in Table 4, the parametrization, the teacher chain and hop partition, the objective, the optimizer, batch size and number of updates, and the inference-time evaluation counts and noise scales; no text data is used. Appendix B.3 specifies the field recipe, the hop parameters, the concurrent-training and finishing schedules, the long-hop targets, and the inference configuration of every row of Tables 5–7. All reported numbers are generative perplexity under GPT-2 Large with a 1024-token context and per-sample unigram entropy in nats, computed on 1000 unconditional samples for the ELF checkpoints and 1024 for LangFlow and the FMLM base from a fixed seed set, and every reported row has been recomputed on a second, disjoint seed set. Noise scales are selected per budget by the rule stated where each comparison is made, and reference samplers are run under their own published protocols. Complete proofs are given in Appendices A.2–A.5. The qualitative examples in Appendix D are unedited outputs of the stated checkpoints at the stated settings. Code, configuration files and the trained Mode B and Mode C parameters will be made publicly available.

REFERENCES

Kendall E Atkinson. *An introduction to numerical analysis*. John wiley & sons, 2008.

Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*, 38: 33346–33382, 2026.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *arXiv preprint arXiv:2406.07507*, 2024.

Jannis Chemseddine, Gregor Kornhardt, and Gabriele Steidl. Spherical flows for sampling categorical data. *arXiv preprint arXiv:2605.05629*, 2026.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. Langflow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Xuanzhao Dong, Wenhui Zhu, Xiwen Chen, Zhipeng Wang, Peijie Qiu, Shao Tang, Xin Li, and Yalin Wang. Llada-medv: Exploring large language diffusion models for biomedical image understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22773–22783, 2026.

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

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

Guangfu Guo, Xiaoqian Lu, Linsey Pang, Weiran Yao, Haolin Chen, Kunpeng Liu, and Long Cheng. Deltaflow: Noise-adaptive bidirectional gated delta networks for embedded language flows. *arXiv preprint arXiv:2608.01240*, 2026a.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, et al. Continuous latent diffusion language model. *arXiv preprint arXiv:2605.06548*, 2026b.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026.

Jaehyeong Jo and Sung Ju Hwang. Continuous diffusion model for language modeling. *Advances in Neural Information Processing Systems*, 38:97394–97430, 2026.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan.github.io/posts/muon*, 6(3):4, 2024.

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

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*, 2026.

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

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36:56998–57025, 2023.

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

Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S Albergo. Discrete flow maps. *arXiv preprint arXiv:2604.09784*, 2026.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. *Proceedings of machine learning research*, 267: 52584, 2025.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

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

Jaehoon Yoo, Wonjung Kim, Floor Eijkelboom, Chanhyuk Lee, Nicholas M Boffi, Seunghoon Hong, and Jinwoo Kim. Self-conditioned flow map language models via fixed-point flows. *arXiv preprint arXiv:2607.00714*, 2026.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. In *International Conference on Learning Representations*, volume 2025, pp. 63186–63227, 2025.

A STATEMENTS AND PROOFS.

A.1 FLOW LANGUAGE MODELS

For completeness, we provide the flow-language-model formulation underlying the affine transport used throughout the main text.

Let V denote the token vocabulary and $y \in V^L$ a token sequence of length L. A representation map $\mathcal{E}$ maps the discrete sequence to a continuous representation $x = \mathcal{E}(y)$, inducing the latent data distribution

$$p(x) = \mathcal{E}_{\#}p(y).$$

Flow language models (FLMs) learn to generate from this distribution through a continuous noisy path

$$z_\rho = \alpha(\rho)x + \sigma(\rho)\varepsilon, \qquad x \sim p(x), \quad \varepsilon \sim \mathcal{N}(0, I), \tag{10}$$

where ρ denotes the native noise-to-data coordinate and $\alpha(\rho), \sigma(\rho)$ specify the interpolation schedule.

Differentiating Eq. 10 gives

$$\frac{dz_\rho}{d\rho} = \dot{\alpha}(\rho)x + \dot{\sigma}(\rho)\varepsilon.$$

For $\sigma(\rho) > 0$, eliminating ε using

$$\varepsilon = \frac{z_\rho - \alpha(\rho)x}{\sigma(\rho)}$$

yields the equivalent x-prediction form

$$v(z_\rho, \rho) = \lambda(\rho)x + \tau(\rho)z_\rho, \tag{11}$$

with

$$\lambda(\rho) = \dot{\alpha}(\rho) - \frac{\alpha(\rho)\dot{\sigma}(\rho)}{\sigma(\rho)}, \qquad \tau(\rho) = \frac{\dot{\sigma}(\rho)}{\sigma(\rho)}.$$

Thus, for fixed (z_ρ, ρ), predicting the velocity is equivalent to predicting the clean representation x up to schedule-dependent coefficients.

Following the x-prediction parameterization used in the main text, let

$$\hat{x} = D_\theta(z_\rho, \rho; c)$$

denote the clean estimate predicted by the denoiser, where c is an optional self-conditioning input. The corresponding flow-matching objective can be written as

$$\mathcal{L}_{\text{FLM}} = \mathbb{E}_{\rho,x,\varepsilon}\left[\|v_\theta(z_\rho, \rho) - v(z_\rho, \rho)\|^2\right] = \mathbb{E}_{\rho,x,\varepsilon}\left[\beta(\rho)\,\|\hat{x} - x\|^2\right], \tag{12}$$

where

$$\beta(\rho) = \lambda(\rho)^2.$$

The exact prediction-space loss may depend on the underlying FLM parameterization, but throughout this work D_θ denotes the full backbone prediction map after any model-specific projection into the continuous state space.

Substituting the learned clean estimate into Eq. 11 gives the associated probability-flow ODE (PF-ODE),

$$\frac{dz_\rho}{d\rho} = \lambda(\rho)D_\theta(z_\rho, \rho; c) + \tau(\rho)z_\rho. \tag{13}$$

The state dependence is linear once the clean estimate is held fixed. Specifically, suppose that a candidate estimate $\hat{x}$ is kept constant over an interval $[\rho, \rho']$. Rearranging Eq. 10, the residual noise coordinate

$$r = \frac{z_\rho - \alpha(\rho)\hat{x}}{\sigma(\rho)}$$

remains fixed under this held-estimate dynamics. Advancing the schedule from ρ to ρ' therefore gives

$$z_{\rho'} = \alpha(\rho')\hat{x} + \sigma(\rho')\frac{z_\rho - \alpha(\rho)\hat{x}}{\sigma(\rho)}.$$

Collecting terms yields

$$z_{\rho'} = \frac{\sigma(\rho')}{\sigma(\rho)} z_\rho + \left(\alpha(\rho') - \frac{\sigma(\rho')}{\sigma(\rho)} \alpha(\rho) \right) \hat{x}. \tag{14}$$

This is precisely the affine transport $\mathcal{P}_{\rho \to \rho'}(z; \hat{x})$ defined in Eq. 1 of the main text.

Importantly, Eq. 14 is exact for a *held-fixed estimate*. It is not an exact solution of the full learned dynamics, since the true denoiser prediction generally changes along the trajectory. The discretization error therefore arises from freezing $\hat{x}$ over the interval rather than from approximating the resulting state update. This property is central to our construction: a candidate estimate determines an entire transported state without requiring an additional denoiser evaluation.

The same transport also applies when the base sampler injects noise between denoiser evaluations. Following the notation of Eq. 2, we write

$$\bar{z}_i = \kappa_i z_i + \eta_i \xi_i, \qquad \xi_i \sim \mathcal{N}(0, I),$$

where κ_i and η_i are determined by the sampling protocol. The re-noised state $\bar{z}_i$ is associated with coordinate $\bar{\rho}_i$, after which the same affine transport $\mathcal{P}_{\bar{\rho}_i \to \rho'}$ is applied. Deterministic sampling is recovered with $\kappa_i = 1$, $\eta_i = 0$, and $\bar{\rho}_i = \rho_i$.

These two properties, x-prediction and closed-form affine transport, are the only FLM-specific ingredients required by our framework. They allow the denoising estimate to influence both the prediction itself and the state at which subsequent predictions are evaluated, which motivates the state–estimate coupling developed in Sec. 3.1.

A.2 Proofs

Proposition 3.1 (Local convergence of joint refinement). *Fix the starting state $\bar{z}_i$, the noise draw ξ_i, and all sampler settings of step i, and write $M_i(\hat{x}) = a_i^{\mathrm{m}} \bar{z}_i + b_i^{\mathrm{m}} \hat{x}$ for the midpoint state in Eq. 4, whose coefficients are read off Eq. 1. Let $\mathcal{U}_i = \{\hat{x} \colon \|\hat{x} - \hat{x}_i^{(0)}\| \le r_i\}$. Assume that D_θ is continuously differentiable in its state and conditioning arguments on a neighborhood of $\{(M_i(\hat{x}), \rho_{m,i}; \hat{x}) \colon \hat{x} \in \mathcal{U}_i\}$, and set*

$$q_i = \sup_{\hat{x} \in \mathcal{U}_i} \left\| b_i^{\mathrm{m}} \partial_z D_\theta + \partial_c D_\theta \right\|_{\mathrm{op}}, \qquad \delta_i = \left\| \mathcal{F}_i(\hat{x}_i^{(0)}) - \hat{x}_i^{(0)} \right\|, \tag{5}$$

with the Jacobians evaluated at $(M_i(\hat{x}), \rho_{m,i}; \hat{x})$ and $\partial_c D_\theta \equiv 0$ for bases without self-conditioning. Vector norms are Euclidean, and the operator norm is induced by the Euclidean norm. If $q_i < 1$ and $\delta_i \le (1 - q_i) r_i$, then $\mathcal{F}_i$ has a unique fixed point $\hat{x}_i^\star \in \mathcal{U}_i$, the iterates remain in $\mathcal{U}_i$, and, for every $K \ge 1$,

$$\left\| \hat{x}_i^{(K)} - \hat{x}_i^\star \right\| \le \frac{q_i^K}{1 - q_i} \delta_i, \qquad \left\| z_{i+1}^{(K)} - z_{i+1}^\star \right\| \le |s_i| \frac{q_i^K}{1 - q_i} \delta_i, \tag{6}$$

where $z_{i+1}^{(K)}$ and $z_{i+1}^\star$ are the states committed from $\hat{x}_i^{(K)}$ and $\hat{x}_i^\star$, and s_i is the estimate coefficient of the step,

$$s_i = \alpha(\rho_{i+1}) - \frac{\sigma(\rho_{i+1})}{\sigma(\bar{\rho}_i)} \alpha(\bar{\rho}_i), \qquad \sigma(\bar{\rho}_i) > 0.$$

Proof. The starting state $\bar{z}_i$, the coordinates $\bar{\rho}_i, \rho_{m,i}, \rho_{i+1}$, and the sampler settings are fixed throughout the refinement.By Eq. 1,

$$\begin{aligned} M_i(\hat{x}) &= \mathcal{P}_{\bar{\rho}_i \to \rho_{m,i}}(\bar{z}_i; \hat{x}) \\ &= \frac{\sigma(\rho_{m,i})}{\sigma(\bar{\rho}_i)} \bar{z}_i + \left(\alpha(\rho_{m,i}) - \frac{\sigma(\rho_{m,i})}{\sigma(\bar{\rho}_i)} \alpha(\bar{\rho}_i) \right) \hat{x}. \end{aligned}$$

Thus $M_i(\hat{x}) = a_i^{\mathrm{m}} \bar{z}_i + b_i^{\mathrm{m}} \hat{x}$, where

$$a_i^{\mathrm{m}} = \frac{\sigma(\rho_{m,i})}{\sigma(\bar{\rho}_i)}, \qquad b_i^{\mathrm{m}} = \alpha(\rho_{m,i}) - \frac{\sigma(\rho_{m,i})}{\sigma(\bar{\rho}_i)} \alpha(\bar{\rho}_i). \tag{15}$$

The assumption $\sigma(\bar{\rho}_i) > 0$ makes these coefficients well defined. Both are fixed throughout the refinement.

Substituting Eq. 15 into Eq. 4 gives

$$\mathcal{F}_i(\hat{x}) = D_\theta\big(a_i^{\mathrm{m}}\bar{z}_i + b_i^{\mathrm{m}}\hat{x}, \rho_{m,i}; c = \hat{x}\big).$$

The candidate estimate affects both the midpoint state and the conditioning argument, while $\rho_{m,i}$ remains fixed. Since

$$\partial_{\hat{x}} M_i(\hat{x}) = b_i^{\mathrm{m}} I, \qquad \partial_{\hat{x}} c = I,$$

where I is the identity on the estimate space, the chain rule yields

$$\begin{aligned}\partial_{\hat{x}}\mathcal{F}_i(\hat{x}) &= \partial_z D_\theta\, \partial_{\hat{x}} M_i(\hat{x}) + \partial_c D_\theta\, \partial_{\hat{x}} c \\ &= b_i^{\mathrm{m}} \partial_z D_\theta + \partial_c D_\theta.\end{aligned} \tag{16}$$

Here $\partial_z D_\theta$ and $\partial_c D_\theta$ are the Jacobians with respect to the state and conditioning arguments, evaluated at $(M_i(\hat{x}), \rho_{m,i}; \hat{x})$. For a backbone without self-conditioning, $\partial_c D_\theta \equiv 0$.

For any $\hat{x}, \hat{x}' \in \mathcal{U}_i$, convexity of $\mathcal{U}_i$ ensures that the segment joining them remains in $\mathcal{U}_i$. The fundamental theorem of calculus gives

$$\mathcal{F}_i(\hat{x}) - \mathcal{F}_i(\hat{x}') = \int_0^1 \partial_{\hat{x}}\mathcal{F}_i\big(\hat{x}' + t(\hat{x} - \hat{x}')\big)(\hat{x} - \hat{x}')\, dt.$$

Using Eq. 16 and the definition of q_i in Eq. 5, we obtain

$$\|\mathcal{F}_i(\hat{x}) - \mathcal{F}_i(\hat{x}')\| \le q_i \|\hat{x} - \hat{x}'\|, \qquad \hat{x}, \hat{x}' \in \mathcal{U}_i. \tag{17}$$

We next establish that $\mathcal{F}_i$ maps $\mathcal{U}_i$ into itself. By Eq. 17 and the definition of δ_i in Eq. 5,

$$\begin{aligned}\|\mathcal{F}_i(\hat{x}) - \hat{x}_i^{(0)}\| &\le \|\mathcal{F}_i(\hat{x}) - \mathcal{F}_i(\hat{x}_i^{(0)})\| + \|\mathcal{F}_i(\hat{x}_i^{(0)}) - \hat{x}_i^{(0)}\| \\ &\le q_i\|\hat{x} - \hat{x}_i^{(0)}\| + \delta_i \\ &\le q_i r_i + \delta_i \le r_i\end{aligned}$$

for every $\hat{x} \in \mathcal{U}_i$. The last inequality follows from $\delta_i \le (1 - q_i) r_i$. Hence $\mathcal{F}_i(\mathcal{U}_i) \subseteq \mathcal{U}_i$. Because $\mathcal{U}_i$ is a closed ball in Euclidean space, it is complete. The Banach fixed-point theorem therefore gives a unique fixed point $\hat{x}_i^\star \in \mathcal{U}_i$. The iterates $\hat{x}_i^{(k+1)} = \mathcal{F}_i(\hat{x}_i^{(k)})$ remain in $\mathcal{U}_i$ and converge to $\hat{x}_i^\star$.

Applying Eq. 17 to successive iterates yields

$$\|\hat{x}_i^{(k+1)} - \hat{x}_i^{(k)}\| \le q_i \|\hat{x}_i^{(k)} - \hat{x}_i^{(k-1)}\|, \qquad k \ge 1.$$

Since $\|\hat{x}_i^{(1)} - \hat{x}_i^{(0)}\| = \delta_i$, induction gives

$$\|\hat{x}_i^{(k+1)} - \hat{x}_i^{(k)}\| \le q_i^k \delta_i, \qquad k \ge 0. \tag{18}$$

Summing Eq. 18 from $k = K$ to the limit gives

$$\begin{aligned}\|\hat{x}_i^{(K)} - \hat{x}_i^\star\| &\le \sum_{k=K}^{\infty} \|\hat{x}_i^{(k+1)} - \hat{x}_i^{(k)}\| \\ &\le \sum_{k=K}^{\infty} q_i^k \delta_i = \frac{q_i^K}{1 - q_i}\, \delta_i.\end{aligned}$$

This proves the estimate bound in Eq. 6.

For the endpoint bound, Eq. 1 gives

$$\mathcal{P}_{\bar{\rho}_i \to \rho_{i+1}}(\bar{z}_i; \hat{x}) = \frac{\sigma(\rho_{i+1})}{\sigma(\bar{\rho}_i)}\bar{z}_i + s_i \hat{x},$$

where

$$s_i = \alpha(\rho_{i+1}) - \frac{\sigma(\rho_{i+1})}{\sigma(\bar{\rho}_i)}\alpha(\bar{\rho}_i).$$

The transport term is identical for the finite iterate and the fixed point. Subtracting the committed states therefore gives

$$z_{i+1}^{(K)} - z_{i+1}^{\star} = s_i\big(\hat{x}_i^{(K)} - \hat{x}_i^{\star}\big).$$

Taking norms and applying the estimate bound yields

$$\begin{aligned}\|z_{i+1}^{(K)} - z_{i+1}^{\star}\| &= |s_i|\,\|\hat{x}_i^{(K)} - \hat{x}_i^{\star}\| \\ &\le |s_i|\frac{q_i^K}{1-q_i}\,\delta_i.\end{aligned}$$

This completes the proof of Eq. 6. □

Corollary 3.2 (Finite-step token stabilization). *Let i denote the final refined step, at which the terminal readout is applied, and adopt the conditions of Proposition 3.1. Let $\mathcal{R}(\hat{x})$ denote the terminal token scores obtained from an estimate through the base's endpoint reconstruction and readout, with all readout settings fixed. Suppose $\mathcal{R}$ is Lipschitz on $\mathcal{U}_i$ with a finite constant L_K, independent of the refinement count K,*

$$\|\mathcal{R}(\hat{x}) - \mathcal{R}(\hat{x}')\|_\infty \le L_K\|\hat{x} - \hat{x}'\|_2, \qquad \hat{x}, \hat{x}' \in \mathcal{U}_i.$$

Here, the infinity norm is taken over all token scores. At position ℓ, let $g_\ell > 0$ be the gap between the largest and second-largest scores in $\mathcal{R}(\hat{x}_i^{\star})$. Then the argmax prediction from $\hat{x}_i^{(K)}$ coincides with the equilibrium prediction whenever

$$\frac{2L_K q_i^K}{1-q_i}\,\delta_i < g_\ell. \tag{7}$$

For a finite sequence with positive margins at every position, the argmax-decoded sequence therefore stabilizes after finitely many refinements.

Proof. By the Lipschitz assumption on $\mathcal{R}$ and Eq. 6,

$$\begin{aligned}\|\mathcal{R}(\hat{x}_i^{(K)}) - \mathcal{R}(\hat{x}_i^{\star})\|_\infty &\le L_K\|\hat{x}_i^{(K)} - \hat{x}_i^{\star}\|_2 \\ &\le \frac{L_K q_i^K}{1-q_i}\,\delta_i.\end{aligned} \tag{19}$$

Since the infinity norm is taken over all positions and all tokens, this bound applies entrywise to every score.

Fix a position ℓ. Write $\mathcal{R}_{\ell,v}(\hat{x})$ for the score of token v at position ℓ, and let $y_\ell^{\star}$ denote the highest-scoring token under $\mathcal{R}(\hat{x}_i^{\star})$. The positive-margin assumption makes this token unique, with

$$g_\ell = \mathcal{R}_{\ell,y_\ell^{\star}}(\hat{x}_i^{\star}) - \max_{v \ne y_\ell^{\star}} \mathcal{R}_{\ell,v}(\hat{x}_i^{\star}) > 0.$$

Consequently, for every competing token $v \ne y_\ell^{\star}$,

$$\mathcal{R}_{\ell,y_\ell^{\star}}(\hat{x}_i^{\star}) - \mathcal{R}_{\ell,v}(\hat{x}_i^{\star}) \ge g_\ell.$$

Adding and subtracting the corresponding equilibrium scores, then applying Eq. 19, gives

$$\begin{aligned}&\mathcal{R}_{\ell,y_\ell^{\star}}(\hat{x}_i^{(K)}) - \mathcal{R}_{\ell,v}(\hat{x}_i^{(K)}) \\ &\quad\ge g_\ell - \big|\mathcal{R}_{\ell,y_\ell^{\star}}(\hat{x}_i^{(K)}) - \mathcal{R}_{\ell,y_\ell^{\star}}(\hat{x}_i^{\star})\big| \\ &\qquad\quad - \big|\mathcal{R}_{\ell,v}(\hat{x}_i^{(K)}) - \mathcal{R}_{\ell,v}(\hat{x}_i^{\star})\big| \\ &\quad\ge g_\ell - 2\|\mathcal{R}(\hat{x}_i^{(K)}) - \mathcal{R}(\hat{x}_i^{\star})\|_\infty \\ &\quad\ge g_\ell - \frac{2L_K q_i^K}{1-q_i}\,\delta_i.\end{aligned} \tag{20}$$

The factor 2 accounts for the possible decrease in the winning score and increase in a competing score. Under Eq. 7, the right-hand side of Eq. 20 is positive for every $v \neq y^\star_\ell$. Thus the argmax prediction at position ℓ equals $y^\star_\ell$.

For a finite sequence with positive margins at every position,

$$\min_\ell g_\ell > 0.$$

Because $0 \leq q_i < 1$ and L_K is finite and independent of the refinement count K,

$$\frac{2L_K q_i^K}{1-q_i}\,\delta_i \longrightarrow 0 \qquad \text{as } K \to \infty.$$

There is therefore an integer K_0 such that Eq. 7 holds at every position for all $K \geq K_0$. The argmax-decoded sequence then equals the equilibrium sequence and remains unchanged under further refinement. □

A.3 Joint refinement and numerical accuracy

Proposition A.1 (Implicit-midpoint interpretation)**.** *Consider the linear interpolant $\alpha(\rho) = \rho$ and $\sigma(\rho) = 1 - \rho$. Fix the re-noised starting state $\bar{z}_i$, the noise draws, and all sampler settings. Let*

$$h_i = \rho_{i+1} - \bar{\rho}_i > 0, \qquad \rho_{m,i} = \frac{\bar{\rho}_i + \rho_{i+1}}{2},$$

with $\bar{\rho}_i < \rho_{i+1} < 1$. Assume the conditions of Proposition 3.1, and let $\hat{x}^\star_i$ be the resulting joint fixed point.
Suppose that, for each (z, ρ) in the region under consideration, the fixed-state equation has a unique solution $D^\star_\theta(z, \rho)$ satisfying

$$D^\star_\theta(z,\rho) = D_\theta\big(z, \rho; D^\star_\theta(z,\rho)\big).$$

For a backbone without self-conditioning, set $D^\star_\theta = D_\theta$. Define the equilibrium velocity field by

$$v^\star_\theta(z,\rho) = \frac{D^\star_\theta(z,\rho) - z}{1-\rho}.$$

Then the converged joint update satisfies

$$z^\star_{i+1} = \bar{z}_i + h_i v^\star_\theta\left(\frac{\bar{z}_i + z^\star_{i+1}}{2}, \rho_{m,i}\right). \tag{21}$$

Converged self-conditioning at the pinned starting pair $(\bar{z}_i, \bar{\rho}_i)$, followed by the same affine transport, instead gives

$$z^{\text{pin}}_{i+1} = \bar{z}_i + h_i v^\star_\theta(\bar{z}_i, \bar{\rho}_i).$$

Let $z^{\text{ODE}}(\rho)$ solve

$$\frac{dz^{\text{ODE}}}{d\rho} = v^\star_\theta(z^{\text{ODE}}(\rho), \rho), \qquad z^{\text{ODE}}(\bar{\rho}_i) = \bar{z}_i,$$

and write $z^{\text{ODE}}_{i+1} = z^{\text{ODE}}(\rho_{i+1})$. Suppose that $v^\star_\theta$ is twice continuously differentiable on a convex neighborhood containing this trajectory and the line segment joining $(\bar{z}_i, \bar{\rho}_i)$ to $(z^\star_{i+1}, \rho_{i+1})$. Assume that the field and its derivatives through order two are uniformly bounded there, with $1 - \rho \geq \epsilon > 0$.
For sufficiently small h_i and every integer $K \geq 1$,

$$\big\|z^{(K)}_{i+1} - z^{\text{ODE}}_{i+1}\big\| \leq C h_i^3 + \frac{h_i}{1-\bar{\rho}_i}\frac{q_i^K}{1-q_i}\,\delta_i, \tag{22}$$

where q_i and δ_i are defined in Eq. 5, and C is independent of K and sufficiently small h_i under the stated uniform bounds.

Proof. Substituting the linear schedules into Eq. 15 gives

$$a_i^{\mathrm{m}} = \frac{1-\rho_{m,i}}{1-\bar{\rho}_i}, \qquad b_i^{\mathrm{m}} = \frac{\rho_{m,i}-\bar{\rho}_i}{1-\bar{\rho}_i} = \frac{h_i}{2(1-\bar{\rho}_i)}.$$

Similarly, the endpoint coefficient in Eq. 6 becomes

$$s_i = \rho_{i+1} - \frac{1-\rho_{i+1}}{1-\bar{\rho}_i}\bar{\rho}_i = \frac{h_i}{1-\bar{\rho}_i}.$$

Thus Eq. 1 yields

$$\begin{aligned} M_i(\hat{x}_i^\star) &= \bar{z}_i + \frac{h_i}{2(1-\bar{\rho}_i)}\big(\hat{x}_i^\star - \bar{z}_i\big), \\ z_{i+1}^\star &= \bar{z}_i + \frac{h_i}{1-\bar{\rho}_i}\big(\hat{x}_i^\star - \bar{z}_i\big). \end{aligned} \tag{23}$$

Comparing these expressions gives

$$M_i(\hat{x}_i^\star) = \frac{\bar{z}_i + z_{i+1}^\star}{2}.$$

By the joint fixed-point equation,

$$\hat{x}_i^\star = D_\theta\big(M_i(\hat{x}_i^\star), \rho_{m,i}; \hat{x}_i^\star\big).$$

Therefore $\hat{x}_i^\star$ solves the fixed-state equation at $(M_i(\hat{x}_i^\star), \rho_{m,i})$. Uniqueness of that solution implies

$$\hat{x}_i^\star = D_\theta^\star\big(M_i(\hat{x}_i^\star), \rho_{m,i}\big).$$

Moreover, Eq. 23 gives

$$\begin{aligned} \hat{x}_i^\star - M_i(\hat{x}_i^\star) &= \left(1 - \frac{h_i}{2(1-\bar{\rho}_i)}\right)\big(\hat{x}_i^\star - \bar{z}_i\big) \\ &= \frac{1-\rho_{m,i}}{1-\bar{\rho}_i}\big(\hat{x}_i^\star - \bar{z}_i\big). \end{aligned}$$

Using the definition of $v_\theta^\star$, we obtain

$$v_\theta^\star\big(M_i(\hat{x}_i^\star), \rho_{m,i}\big) = \frac{\hat{x}_i^\star - \bar{z}_i}{1-\bar{\rho}_i}.$$

Substitution into Eq. 23 proves Eq. 21.

At the pinned starting pair, the converged estimate is $D_\theta^\star(\bar{z}_i, \bar{\rho}_i)$. Applying Eq. 1 with this estimate gives

$$\begin{aligned} z_{i+1}^{\mathrm{pin}} &= \bar{z}_i + \frac{h_i}{1-\bar{\rho}_i}\big(D_\theta^\star(\bar{z}_i, \bar{\rho}_i) - \bar{z}_i\big) \\ &= \bar{z}_i + h_i v_\theta^\star(\bar{z}_i, \bar{\rho}_i), \end{aligned}$$

which is the explicit Euler update for the same field.

We next bound the local error of the converged joint update. Differentiating the ODE along its solution gives

$$\frac{d^2 z^{\mathrm{ODE}}}{d\rho^2} = \partial_\rho v_\theta^\star + \partial_z v_\theta^\star\, v_\theta^\star,$$

where the terms on the right are evaluated at $(z^{\mathrm{ODE}}(\rho), \rho)$. Taylor expansion about the starting pair therefore yields

$$z_{i+1}^{\mathrm{ODE}} = \bar{z}_i + h_i v_\theta^\star(\bar{z}_i, \bar{\rho}_i) + \frac{h_i^2}{2}\left[\partial_\rho v_\theta^\star + \partial_z v_\theta^\star\, v_\theta^\star\right]_{(\bar{z}_i, \bar{\rho}_i)} + O(h_i^3).$$

In particular,

$$\frac{\bar{z}_i + z_{i+1}^{\mathrm{ODE}}}{2} = \bar{z}_i + \frac{h_i}{2} v_\theta^\star(\bar{z}_i, \bar{\rho}_i) + O(h_i^2).$$

Expanding the velocity at this averaged state and at $\rho_{m,i} = \bar{\rho}_i + h_i/2$ gives

$$v_\theta^\star\left(\frac{\bar{z}_i + z_{i+1}^{\mathrm{ODE}}}{2}, \rho_{m,i}\right) = v_\theta^\star(\bar{z}_i, \bar{\rho}_i) + \frac{h_i}{2}\left[\partial_\rho v_\theta^\star + \partial_z v_\theta^\star\, v_\theta^\star\right]_{(\bar{z}_i, \bar{\rho}_i)} + O(h_i^2).$$

Comparing the two expansions gives

$$\begin{aligned} z_{i+1}^{\mathrm{ODE}} &= \bar{z}_i + h_i v_\theta^\star\left(\frac{\bar{z}_i + z_{i+1}^{\mathrm{ODE}}}{2}, \rho_{m,i}\right) + r_i^{\mathrm{ODE}}, \\ \|r_i^{\mathrm{ODE}}\| &\le C_0 h_i^3. \end{aligned} \tag{24}$$

The uniform smoothness bounds make C_0 independent of sufficiently small h_i.

Let L bound $\|\partial_z v_\theta^\star\|_{\mathrm{op}}$ on the stated neighborhood. Convexity then gives a state Lipschitz bound between the midpoint arguments in Eqs. 21 and 24. Subtracting these equations yields

$$\|z_{i+1}^{\mathrm{ODE}} - z_{i+1}^\star\| \le \frac{h_i L}{2}\|z_{i+1}^{\mathrm{ODE}} - z_{i+1}^\star\| + C_0 h_i^3.$$

For $h_i L \le 1$, rearranging gives

$$\|z_{i+1}^{\mathrm{ODE}} - z_{i+1}^\star\| \le \frac{C_0 h_i^3}{1 - h_i L/2} \le 2C_0 h_i^3. \tag{25}$$

Finally, the triangle inequality gives

$$\begin{aligned} \|z_{i+1}^{(K)} - z_{i+1}^{\mathrm{ODE}}\| &\le \|z_{i+1}^{(K)} - z_{i+1}^\star\| + \|z_{i+1}^\star - z_{i+1}^{\mathrm{ODE}}\| \\ &\le \frac{h_i}{1-\bar{\rho}_i}\frac{q_i^K}{1-q_i}\,\delta_i + 2C_0 h_i^3, \end{aligned}$$

where the last inequality uses Eq. 6, $s_i = h_i/(1-\bar{\rho}_i)$, and Eq. 25. Taking $C = 2C_0$ proves Eq. 22. □

Equation 22 separates transport discretization error from incomplete refinement. For the linear interpolant and an arithmetic midpoint in ρ, the converged joint update is an implicit-midpoint step of the equilibrium field. Converged refinement at the pinned starting pair instead gives an explicit Euler step of that field.

Under the stated uniform bounds, a sufficient condition for the finite-refinement update to retain the $O(h_i^3)$ local endpoint error bound is

$$\frac{q_i^K}{1-q_i}\,\delta_i = O(h_i^2) \qquad \text{as } h_i \to 0,$$

where K may depend on h_i. This condition relates the initial residual and refinement depth to numerical accuracy without assuming that one Picard evaluation is sufficient. The result concerns a deterministic transport substep of the learned equilibrium field. It does not establish the accuracy order of the complete re-noised sampler or guarantee improved generation metrics.

A.4 Error propagation in hop distillation

The exact segment decomposition separates the known transport and noise terms from the learned content. The following proposition relates errors in the predicted content and outgoing estimate to errors across a sequence of student hops.

Proposition A.2 (Stability of feedback-preserving hops)**.** *Fix the teacher, the sampler settings, and the noise draws for J segments. Teacher and student use the same coefficients A_j and accumulated noise N_j from Eq. 8. For each segment j, let $G_j^T(z, \hat{x})$ and $E_j^T(z, \hat{x})$ denote the content aggregate and outgoing estimate obtained by continuing the teacher from the boundary*

$(z, \hat{x})$. *The teacher rollout therefore satisfies*

$$\begin{aligned} z^T_{T_{j+1}} &= A_j z^T_{T_j} + N_j + G^T_j(z^T_{T_j}, \hat{x}^T_{T_j}), \\ \hat{x}^T_{T_{j+1}} &= E^T_j(z^T_{T_j}, \hat{x}^T_{T_j}). \end{aligned} \tag{26}$$

Assume that $(z, \hat{x})$ contains the boundary information required to continue the teacher segment. Suppose that, for all boundary pairs in a region containing both rollouts,

$$\begin{aligned} \|G^T_j(z, \hat{x}) - G^T_j(z', \hat{x}')\| &\le L_{Gz,j}\|z - z'\| + L_{Gx,j}\|\hat{x} - \hat{x}'\|, \\ \|E^T_j(z, \hat{x}) - E^T_j(z', \hat{x}')\| &\le L_{Ez,j}\|z - z'\| + L_{Ex,j}\|\hat{x} - \hat{x}'\|, \end{aligned} \tag{27}$$

where the four constants are finite and nonnegative. All vector norms are Euclidean.
Let $(\hat{G}_j, \hat{x}^S_{T_{j+1}})$ be the student outputs after its prescribed hop evaluations, with committed state

$$z^S_{T_{j+1}} = A_j z^S_{T_j} + N_j + \hat{G}_j.$$

Define the local defects relative to the teacher continued from the student boundary by

$$\begin{aligned} d_{G,j} &= \big\|\hat{G}_j - G^T_j(z^S_{T_j}, \hat{x}^S_{T_j})\big\|, \\ d_{x,j} &= \big\|\hat{x}^S_{T_{j+1}} - E^T_j(z^S_{T_j}, \hat{x}^S_{T_j})\big\|. \end{aligned} \tag{28}$$

Let

$$\mathbf{e}_j = \begin{pmatrix} \|z^S_{T_j} - z^T_{T_j}\| \\ \|\hat{x}^S_{T_j} - \hat{x}^T_{T_j}\| \end{pmatrix}, \qquad \mathbf{d}_j = \begin{pmatrix} d_{G,j} \\ d_{x,j} \end{pmatrix}.$$

Then, componentwise,

$$\mathbf{e}_{j+1} \le \mathbf{L}_j \mathbf{e}_j + \mathbf{d}_j, \qquad \mathbf{L}_j = \begin{pmatrix} |A_j| + L_{Gz,j} & L_{Gx,j} \\ L_{Ez,j} & L_{Ex,j} \end{pmatrix}. \tag{29}$$

If teacher and student share their initial state and estimate, then

$$\mathbf{e}_J \le \sum_{j=0}^{J-1} \mathbf{L}_{J-1} \cdots \mathbf{L}_{j+1} \mathbf{d}_j, \tag{30}$$

where an empty matrix product is the identity.

Proof. Subtracting the teacher state update in Eq. 26 from the student update cancels the shared noise term,

$$z^S_{T_{j+1}} - z^T_{T_{j+1}} = A_j(z^S_{T_j} - z^T_{T_j}) + \hat{G}_j - G^T_j(z^T_{T_j}, \hat{x}^T_{T_j}).$$

Insert the teacher content evaluated at the student boundary,

$$\hat{G}_j - G^T_j(z^T_{T_j}, \hat{x}^T_{T_j}) = \hat{G}_j - G^T_j(z^S_{T_j}, \hat{x}^S_{T_j}) + G^T_j(z^S_{T_j}, \hat{x}^S_{T_j}) - G^T_j(z^T_{T_j}, \hat{x}^T_{T_j}).$$

The first difference is the local content defect. The second arises from the difference between the incoming teacher and student boundaries. Taking norms and applying Eqs. 27 and 28 gives

$$\begin{aligned} \|z^S_{T_{j+1}} - z^T_{T_{j+1}}\| &\le |A_j|\|z^S_{T_j} - z^T_{T_j}\| + d_{G,j} + L_{Gz,j}\|z^S_{T_j} - z^T_{T_j}\| + L_{Gx,j}\|\hat{x}^S_{T_j} - \hat{x}^T_{T_j}\| \\ &= (|A_j| + L_{Gz,j})\|z^S_{T_j} - z^T_{T_j}\| + L_{Gx,j}\|\hat{x}^S_{T_j} - \hat{x}^T_{T_j}\| + d_{G,j}. \end{aligned}$$

This is the first component of Eq. 29.

For the outgoing estimate, use Eq. 26 and insert the teacher estimate evaluated at the student boundary,

$$\hat{x}^S_{T_{j+1}} - \hat{x}^T_{T_{j+1}} = \hat{x}^S_{T_{j+1}} - E^T_j(z^S_{T_j}, \hat{x}^S_{T_j}) + E^T_j(z^S_{T_j}, \hat{x}^S_{T_j}) - E^T_j(z^T_{T_j}, \hat{x}^T_{T_j}).$$

The triangle inequality and Eqs. 27 and 28 yield

$$\|\hat{x}^S_{T_{j+1}} - \hat{x}^T_{T_{j+1}}\| \le L_{Ez,j}\|z^S_{T_j} - z^T_{T_j}\| + L_{Ex,j}\|\hat{x}^S_{T_j} - \hat{x}^T_{T_j}\| + d_{x,j}.$$

This proves the second component and hence Eq. 29.

Every entry of $\mathbf{L}_j$ is nonnegative, so multiplication by $\mathbf{L}_j$ preserves componentwise inequalities. Repeated substitution gives

$$\mathbf{e}_J \le \mathbf{L}_{J-1} \cdots \mathbf{L}_0 \mathbf{e}_0 + \sum_{j=0}^{J-1} \mathbf{L}_{J-1} \cdots \mathbf{L}_{j+1} \mathbf{d}_j.$$

The shared initial boundary gives $\mathbf{e}_0 = 0$, which proves Eq. 30. □

Connection to the hop objective. When the teacher is continued from the student boundary, both endpoints contain the same transport and noise terms, $A_j z^S_{T_j} + N_j$. Their difference is therefore exactly the difference between their content aggregates. Thus $d_{G,j}$ measures the local endpoint error without an approximation to the known transport or noise.

Equation 29 also separates the roles of the two predicted outputs. The content defect contributes directly to the next state error, while the estimate defect enters the carried feedback and can influence subsequent states through $L_{Gx,j}$. Supervising both outputs therefore targets the two components of the rollout error.

For the ELF-family students, whose hop loss is the sum of squared errors on the content aggregate and on the outgoing estimate, the two loss terms at a student boundary are $d^2_{G,j}$ and $d^2_{x,j}$ up to normalization. For students whose committed update carries the estimate inside the affine step, the endpoint-state loss is the squared defect of the committed increment, which is the first row of Eq. 29 in that parametrization. On-policy teacher continuation supplies targets at the boundaries where these defects are defined. The analysis applies to the implemented finite teacher chain, so teacher targets may be generated with a prescribed finite number of refinement evaluations.

Boundary memory. For configurations that extrapolate from successive estimates, the boundary state is augmented with the preceding estimate. Applying the same componentwise argument to this augmented state accounts for the additional feedback dependence. The teacher continuation and local defects are defined using the boundary information and memory conventions of the corresponding configuration.

Proof of convergence in flow map distillation

A.5 Convergence of the student's hop iteration

For Mode B's endpoint-input parameterization, repeated evaluations of a trained hop define a fixed-point iteration on the carried estimate. The incoming state, noise draws, and sampler settings remain fixed, while the estimate updates the surrogate input and any estimate-conditioning interface used by the student. The following proposition establishes convergence of this inference-time iteration under a contraction assumption. It complements the fine-step analysis in Proposition 3.1.

Proposition A.3 (Convergence of hop iteration)**.** *Fix the trained student, the incoming state z_{T_j}, and all noise draws and sampler settings for hop j. Define the content and estimate maps by*

$$\big(\hat{G}_j(\hat{x}), \mathcal{E}_j(\hat{x})\big) = H_\phi\big(A_j z_{T_j} + N_j + V_j \hat{x}, \rho_{T_{j+1}}; c = \hat{x}\big), \tag{31}$$

where $V_j = \sum_{i=T_j}^{T_{j+1}-1} V_{ji}$ is the aggregate estimate coefficient from Eq. 8. The quantities A_j, N_j, V_j, and $\rho_{T_{j+1}}$ remain fixed across evaluations. The conditioning argument is omitted for students without an estimate-conditioning interface.
Let $\mathcal{X}_j$ be a nonempty closed subset of the Euclidean estimate space, and assume $\mathcal{E}_j(\mathcal{X}_j) \subseteq \mathcal{X}_j$. Suppose that, for all $\hat{x}, \hat{x}' \in \mathcal{X}_j$,

$$\begin{aligned} \|\mathcal{E}_j(\hat{x}) - \mathcal{E}_j(\hat{x}')\| &\le q_j^S \|\hat{x} - \hat{x}'\|, \\ \|\hat{G}_j(\hat{x}) - \hat{G}_j(\hat{x}')\| &\le L_{G,j} \|\hat{x} - \hat{x}'\|, \end{aligned} \tag{32}$$

where $0 \le q_j^S < 1$, $0 \le L_{G,j} < \infty$, and all vector norms are Euclidean. Starting from $\hat{x}^{(0)} \in \mathcal{X}_j$, define

$$\hat{x}^{(k+1)} = \mathcal{E}_j(\hat{x}^{(k)}), \qquad k \ge 0.$$

Then the iterates remain in $\mathcal{X}_j$ and converge to the unique fixed point $\hat{x}_j^\star$ of $\mathcal{E}_j$ in $\mathcal{X}_j$. For every integer $K \ge 1$,

$$\|\hat{x}^{(K)} - \hat{x}_j^\star\| \le \frac{(q_j^S)^K}{1 - q_j^S}\|\hat{x}^{(1)} - \hat{x}^{(0)}\|. \tag{33}$$

Define the committed and equilibrium endpoints by

$$\hat{z}_{T_{j+1}}^{(K)} = A_j z_{T_j} + N_j + \hat{G}_j(\hat{x}^{(K-1)}),$$
$$\hat{z}_{T_{j+1}}^{\star} = A_j z_{T_j} + N_j + \hat{G}_j(\hat{x}_j^\star).$$

They satisfy

$$\begin{aligned}\|\hat{z}_{T_{j+1}}^{(K)} - \hat{z}_{T_{j+1}}^{\star}\| &\le L_{G,j}\|\hat{x}^{(K-1)} - \hat{x}_j^\star\| \\ &\le \frac{L_{G,j}}{1 - q_j^S}\|\hat{x}^{(K)} - \hat{x}^{(K-1)}\|.\end{aligned} \tag{34}$$

Proof. The set $\mathcal{X}_j$ is closed in Euclidean space and is therefore complete. By Eq. 32, $\mathcal{E}_j$ is a contraction. The assumption $\mathcal{E}_j(\mathcal{X}_j) \subseteq \mathcal{X}_j$ ensures that every iterate remains in $\mathcal{X}_j$. The Banach fixed-point theorem then gives a unique fixed point $\hat{x}_j^\star \in \mathcal{X}_j$ and convergence from the initial estimate $\hat{x}^{(0)}$.

Applying Eq. 32 to successive iterates gives

$$\begin{aligned}\|\hat{x}^{(k+1)} - \hat{x}^{(k)}\| &= \|\mathcal{E}_j(\hat{x}^{(k)}) - \mathcal{E}_j(\hat{x}^{(k-1)})\| \\ &\le q_j^S\|\hat{x}^{(k)} - \hat{x}^{(k-1)}\|, \qquad k \ge 1.\end{aligned}$$

Induction therefore yields

$$\|\hat{x}^{(k+1)} - \hat{x}^{(k)}\| \le (q_j^S)^k\|\hat{x}^{(1)} - \hat{x}^{(0)}\|, \qquad k \ge 0. \tag{35}$$

Summing Eq. 35 from the K-th iterate to the limit gives

$$\begin{aligned}\|\hat{x}^{(K)} - \hat{x}_j^\star\| &\le \sum_{k=K}^{\infty}\|\hat{x}^{(k+1)} - \hat{x}^{(k)}\| \\ &\le \sum_{k=K}^{\infty}(q_j^S)^k\|\hat{x}^{(1)} - \hat{x}^{(0)}\| \\ &= \frac{(q_j^S)^K}{1 - q_j^S}\|\hat{x}^{(1)} - \hat{x}^{(0)}\|.\end{aligned}$$

This proves Eq. 33.

By Eq. 31, the K-th evaluation receives $\hat{x}^{(K-1)}$ and returns both $\hat{G}_j(\hat{x}^{(K-1)})$ and $\hat{x}^{(K)}$. Subtracting the committed and equilibrium endpoints cancels their shared transport and noise terms,

$$\hat{z}_{T_{j+1}}^{(K)} - \hat{z}_{T_{j+1}}^{\star} = \hat{G}_j(\hat{x}^{(K-1)}) - \hat{G}_j(\hat{x}_j^\star).$$

The content Lipschitz bound in Eq. 32 therefore gives

$$\|\hat{z}_{T_{j+1}}^{(K)} - \hat{z}_{T_{j+1}}^{\star}\| \le L_{G,j}\|\hat{x}^{(K-1)} - \hat{x}_j^\star\|.$$

To express this bound through the last observed update, use $\mathcal{E}_j(\hat{x}_j^\star) = \hat{x}_j^\star$ and $\mathcal{E}_j(\hat{x}^{(K-1)}) = \hat{x}^{(K)}$. The triangle inequality gives

$$\begin{aligned}\|\hat{x}^{(K-1)} - \hat{x}_j^\star\| &\le \|\hat{x}^{(K-1)} - \hat{x}^{(K)}\| \\ &\quad + \|\mathcal{E}_j(\hat{x}^{(K-1)}) - \mathcal{E}_j(\hat{x}_j^\star)\| \\ &\le \|\hat{x}^{(K)} - \hat{x}^{(K-1)}\| + q_j^S\|\hat{x}^{(K-1)} - \hat{x}_j^\star\|.\end{aligned}$$

Since $q_j^S < 1$, rearranging yields

$$\|\hat{x}^{(K-1)} - \hat{x}_j^\star\| \le \frac{1}{1 - q_j^S} \|\hat{x}^{(K)} - \hat{x}^{(K-1)}\|.$$

Combining the last two inequalities proves Eq. 34. □

The proposition establishes convergence of both the carried estimate and the committed endpoint to the trained hop's equilibrium. The index $K-1$ reflects that the committed content is computed from the estimate entering the final evaluation. Under the stated contraction and Lipschitz assumptions, Eq. 34 provides an a posteriori endpoint-error bound in terms of the last estimate update. Agreement with the teacher is addressed separately by the local defects and rollout bound in Proposition A.2.

B Implementation Details

Mode A uses the released checkpoints of ELF-B/M/L Hu et al. (2026), LangFlow Chen et al. (2026), and the one-hot flow language model (FMLM-base) Lee et al. (2026) without modifying their architecture, weights, or readout. Mode B initializes trainable students from the corresponding released checkpoints, with the hop-specific components described below. Mode C trains a new ELF-B field and protected hop parameters. The pretrained encoders and tokenizers are retained.

Sequences have a generation length of 1024 tokens. Generative perplexity is measured with GPT-2 Large Radford et al. (2019). Entropy is the mean per-sequence unigram token entropy in nats, computed using the evaluation tokenizer. Rep-4 is the mean within-sequence fraction of repeated whitespace-delimited word 4-grams. ELF evaluations use 1000 samples, while LangFlow Chen et al. (2026) and the FMLM-base Lee et al. (2026) use 1024 samples. Evaluations use a fixed record seed set, with a disjoint second seed set reported where stated. LangFlow Lee et al. (2026) perplexity uses its native evaluation mask, including valid EOS tokens.

ELF sampling uses bf16 autocast, following its evaluation loop. LangFlow uses float64 transport arithmetic with a mixed-precision backbone, while the one-hot model uses fp32 states and bf16 autocast for the backbone blocks. The same precision conventions are used for the corresponding reproduced baselines. ELF uses a separate decoder evaluation. LangFlow Mode A includes a final readout evaluation, whereas LangFlow Mode B and the FMLM-base read tokens from the final prediction without an additional decoder evaluation.

B.1 Mode A: Training-Free Sampling

Mode A applies the joint refinement of Section 3 to a released checkpoint without any training. This appendix specifies the chain on each family and the exact configuration behind every row of Tab. 3.

Grid. For ELF Hu et al. (2026), the chain integrates the front region of the interpolant, $t < 0.35$, with K fine steps and then applies the standard tail defined below. The front points are taken at quantile-spaced indices of a frozen logit-normal grid drawn once with seed 42, using $P_{\mathrm{mean}} = -0.8$ on ELF-B and ELF-M and $P_{\mathrm{mean}} = -1.5$ on ELF-L, with $P_{\mathrm{std}} = 0.8$; for chains of 28 or more fine steps the base grid is drawn with 64 points so that the front region is sufficiently resolved. With one initial estimate, one corrector evaluation per fine step and the standard tail, a chain of K fine steps costs $K + 3$ NFE; each additional refinement evaluation adds one.

Readouts. Every ELF Hu et al. (2026) configuration leaves the front region at its last fine step and reads tokens in one of three ways. The *standard tail* completes the trajectory with one of ELF's hybrid SDE steps and one Euler step and then applies the decoder; it adds two network evaluations before the decoder call and is the readout of the official sampler and of every Mode A row. The *decoder-only* readout applies the frozen decoder directly to the final estimate and adds no evaluation. The *refine* readout applies one corrector evaluation of the base to the committed state and then the decoder, adding one. Throughout the paper, NFE counts network evaluations without the final decoder call. LangFlow and the one-hot FLM read tokens from the final prediction and have no tail.

One step. At step i the state is re-noised by ELF's hybrid rule,

$$\bar{z}_i = \kappa_i z_i + \eta_i \xi_i, \qquad \kappa_i = \mathrm{clip}(1 - \gamma h_i, 0, 1), \qquad \eta_i = (1 - \kappa_i)\, s,$$

where $h_i = \rho_{i+1} - \rho_i$, s is the checkpoint's denoising noise scale and $\xi_i \sim \mathcal{N}(0, I)$; the coordinate moves back to $\bar{\rho}_i = \kappa_i \rho_i$, and $\gamma = 0$ recovers deterministic sampling. The initial estimate is extrapolated from the two previous estimates,

$$\hat{x}_i^{(0)} = \hat{x}_{i-1} + \omega\,(\hat{x}_{i-1} - \hat{x}_{i-2}),$$

with $\hat{x}_{i-1}$ alone when only one previous estimate exists; at the first step a single predictor evaluation with a zero self-conditioning input provides it. One joint corrector evaluation follows at $\rho_{m,i} = \frac{1}{2}(\bar{\rho}_i + \rho_{i+1})$, at the state transported to the midpoint with $\hat{x}_i^{(0)}$, which also supplies the self-conditioning input. The step is completed by the affine transport of $\bar{z}_i$ to ρ_{i+1} with the refined estimate. All ELF configurations use $\omega = 0.5$ and classifier free guidance Ho & Salimans (2022), $w = 3.0$.

Placed second evaluations. A configuration may add a second corrector evaluation at a small number of fine steps. The steps are chosen once, before evaluation, by running the one-evaluation chain on a calibration set of sequences disjoint from the evaluation seeds and selecting the fine steps at which the corrector moves the estimate most, measured by the relative change of the estimate at that step. Each placed evaluation adds one NFE and leaves the grid, the noise scale and all other settings unchanged. The two 32-NFE configurations on ELF-B and ELF-L use two placed evaluations; no other row uses any.

LangFlow. LangFlow Chen et al. (2026) uses its native sampling grid, with the midpoint evaluation taken in its log-SNR conditioning coordinate. The reported configurations use two evaluations per fine step and extrapolation coefficient $\omega = 1.5$. Tokens are read by one final network evaluation at the terminal grid point, which is not counted, so a K-step chain costs $2K$ NFE. The deterministic rows apply no re-noising on either side; the re-noised rows apply the exact-marginal re-injection rule of the family's own schedule identically to both samplers, with γ selected per budget on each side.

The one-hot base. The FMLM base Lee et al. (2026) has no self-conditioning input, so the estimate re-enters the computation only through the transported state. The chain uses the model's native conditioning coordinate for its grid and midpoint, extrapolation coefficient $\omega = 0.5$, and two evaluations per fine step, so a K-step chain costs $2K$ NFE; tokens are obtained by argmax from the final prediction without an additional evaluation. Both samplers apply the same re-noising rule, $\bar{z}_i = \frac{t_b}{t}\, z_i + \eta_i \xi_i$ with $t_b = \max\{t_{\min},\, t - \gamma(t_{i+1} - t)\}$, where $t_{\min}$ is the initial interpolant time and $\xi_i \sim \mathcal{N}(0, I)$ is fresh noise; we set $\eta_i^2 = (1 - t_b)^2 - (t_b/t)^2(1 - t)^2$, clamped below at zero, so that $\bar{z}_i$ follows the prescribed conditional Gaussian marginal of the linear interpolant. Within a configuration γ is constant across steps and samples, and re-noising is omitted at the initial step.

Noise re-injection protocol. ELF varies its re-injection scale with the sampling budget. We follow the same protocol on both sides of every comparison: weights and guidance are fixed, and γ is selected per budget for the reference and for our chain alike. Generative perplexity and entropy are reported jointly so that the resulting quality–diversity operating points can be compared; published reference numbers retain their reported settings.

Reported operating points. Table 3 pairs each reference budget with the Mode A configuration that reaches it. On the ELF checkpoints, the 19-NFE configuration runs 16 fine steps, and the 32-NFE configuration runs 27 fine steps with two placed second evaluations on ELF-B and ELF-L and 29 fine steps without them on ELF-M; every ELF row ends with the standard tail. On LangFlow and the one-hot base, every configuration uses two evaluations per fine step, so the NFE column gives twice the number of fine steps. The reference in each row is the original sampler of the same checkpoint under the same protocol: the ELF sampler at its best-gPPL noise scale, the deterministic LangFlow sampler at its published values, and the re-noised LangFlow and one-hot samplers at the largest noise scale whose entropy remains within the band of their deterministic default. Our noise scale is selected per budget by the same rule, with weights and guidance held fixed, and the selected values are those listed in the table.

B.2 MODE B: FEEDBACK-PRESERVING HOP DISTILLATION

Student. On ELF, the student is a trainable copy of the released denoiser with one new head. This head copies the final prediction layer and reads the trunk representation to emit the outgoing estimate $\hat{x}_{T_{j+1}}$. The content aggregate is parameterized through the base's prediction head, $\hat{G}_j = V_j h_\phi(\cdot)$. At initialization, both prediction heads therefore reproduce the base prediction at the endpoint guess

$$\tilde{z}_{T_{j+1}} = A_j z_{T_j} + N_j + V_j \hat{x}_{T_j},$$

conditioned at $\rho_{T_{j+1}}$. The hop constants A_j, B_{ji} and V_{ji} follow from the chain grid and re-noising schedule in Eq. 8.

On LangFlow, the main head supplies the outgoing estimate through the model's native token-to-embedding prediction map. A zero-initialised residual head supplements the contribution $V_j \hat{x}_{T_{j+1}}$; their sum forms the content aggregate. The incoming estimate enters through the native self-conditioning input. The state loss supervises the same committed state used at inference. The learned schedule parameters remain frozen, and dropout is disabled during distillation.

On the one-hot base, the incoming token probability distribution is fed through a zero-initialised projection added to the trunk. The main head predicts the content distribution, and a second head predicts the outgoing estimate distribution. The four-hop student uses the endpoint guess as input. The two-hop student instead uses an affine midpoint guess and its corresponding conditioning coordinate; the final state is still committed using the full-hop affine constants.

Teacher chains and hop layouts. Teacher targets are produced online from the frozen base; no text data is used. At every update, fresh noise is sampled and propagated through the base's own chain. On ELF-B the chain has 12 fine steps with $\gamma = 2.0$ and guidance $w = 3.0$, each step being a predictor evaluation followed by one joint corrector evaluation; the four-hop student covers it with segments of 6, 3, 2 and 1 steps, and the one-hop student with a single 12-step hop. On ELF-M the chain has 16 fine steps with $\gamma = 1.0$ and $w = 1.5$, and the four-hop student uses segments of 8, 4, 3 and 1 steps. On the one-hot base the chain is the deterministic 64×2 chain of Mode A with extrapolation coefficient 0.5; the four-hop student uses four 16-step segments and the two-hop student two 32-step segments. On LangFlow the chain is the deterministic 32×2 chain with log-SNR midpoint evaluations and extrapolation coefficient 1.0; the eight-hop student uses eight 4-step segments.

Losses and optimization. On ELF, the loss is the squared error of the predicted content aggregate against the teacher's content aggregate, equivalently the squared error of the committed endpoint, plus an equally weighted squared error on the outgoing estimate; squared errors are summed over representation dimensions and averaged over positions and samples. On LangFlow and the one-hot base, the estimate loss is $\mathrm{KL}(p_{\mathrm{teacher}} \,\|\, p_{\mathrm{student}})$, while the state loss remains squared error.

On half of the optimization updates, hop inputs are the student's own free-running boundary states, and targets are obtained by continuing the teacher from those states with the corresponding noise draws; the remaining updates use teacher boundary states.

Feedback supervision re-evaluates a hop using its own detached outgoing estimate to rebuild the surrogate input; the first evaluation and the final supervised feedback evaluation are regressed to the same targets. The ELF students supervise the third iterate on alternate updates, the one-hop student after a 1000-update warm-up at depth one. The one-hot four-hop student supervises the second iterate throughout, the one-hot two-hop student after a 1000-update warm-up, and the LangFlow student the second iterate on every update.

For ELF students, the incoming estimate is extrapolated from the third hop onward as $\hat{x}_{T_j} + \beta(\hat{x}_{T_j} - \hat{x}_{T_{j-1}})$ with $\beta = 0.5$, in training and at inference; this inter-hop extrapolation is distinct from the teacher's within-chain extrapolation.

Every student is trained in one stage from the released base with AdamW, $(\beta_1, \beta_2) = (0.9, 0.999)$, a cosine learning rate from 10^{-4} to 10^{-5}, zero weight decay and gradient clipping at 1.0. The ELF-B students take $20\,000$ updates at global batch 64, the ELF-M student $20\,000$ updates at batch 32, the one-hot students 4000 updates at batch 8, and the LangFlow student 4000 updates at batch 8. All reported rows use the raw end-of-stage weights.

Inference. Each hop can be evaluated multiple times. The returned estimate is fed back through the surrogate input and the family's estimate-conditioning interface, while the hop's starting state and noise draws remain fixed; the state is committed after the final evaluation. A k-hop student with evaluation counts $(p_1, \ldots, p_k)$ therefore costs $\sum_j p_j$ NFE plus the readout of Appendix B.1. The ELF rows of Tab. 4 use the decoder-only readout, and the LangFlow and one-hot rows read tokens from the final hop prediction, so NFE equals the number of hop evaluations in every row.

The rows of Tab. 4 are obtained as follows. On ELF-B the one-hop student is run with one evaluation at $\gamma = 1.5$ (1 NFE); the four-hop student with one evaluation per hop at $\gamma = 1.75$ (4 NFE), two evaluations per hop at $\gamma = 2.25$ (8 NFE), and three evaluations per hop at $\gamma = 2.25$ (12 NFE), always with $w = 3.0$. On ELF-M the four-hop student is run at $\gamma = 2.0$ and $w = 1.5$ with one evaluation per hop (4 NFE) and with evaluation counts $(1, 2, 2, 2)$ (7 NFE). For stochastic ELF rollouts the hop constants are recomputed from the inference-time γ with fixed student weights and guidance, following the reference's practice of selecting the re-injection per budget. On the one-hot base the two-hop student is run with one evaluation per hop (2 NFE) and the four-hop student with one and three evaluations per hop (4 and 12 NFE); on LangFlow the eight-hop student is run with two, three and four evaluations per hop (16, 24 and 32 NFE). The one-hot and LangFlow configurations use no noise re-injection.

B.3 Mode C: joint field and hop learning

Field. The field is a new ELF-B denoiser trained from random initialization on OpenWebText Gokaslan & Cohen (2019) using the ELF Hu et al. (2026) training recipe and a frozen T5-small encoder Raffel et al. (2020). Training uses Muon Jordan et al. (2024) with auxiliary Nesterov-Adam updates for non-matrix parameters and zero weight decay. The learning rate increases linearly to 2×10^{-3} over 9508 warm-up updates and remains constant thereafter. The global batch size is 512, and training runs for five epochs, corresponding to $95\,085$ updates. The field retains ELF's training objective, comprising velocity-matching squared error on denoising examples and token cross-entropy on decoder examples. Its evaluation EMA has decay 0.9999.

Hop parameters. The hop model is the field itself with protected parameters ϕ active only in hop mode. These comprise rank-64 residual adapters on the attention QKV and output projections and the two MLP projections in every transformer block, a second copy of the final prediction layer for the outgoing estimate, a zero-initialised noise projection, and a hop-length embedding. Together, these components contain approximately 11.9M parameters. With all hop-specific components disabled, the network recovers the field. The hop loss does not update the field parameters θ.

Concurrent training. Hop updates begin at field update $10\,000$ and run once per four field updates, using global batch size 64, AdamW Loshchilov & Hutter (2017) with zero weight decay, a cosine learning rate from 10^{-4} to 10^{-5} over a horizon of $21\,000$ hop updates, and gradient clipping at 1.0. Targets are produced online by the twelve-step joint-refinement teacher chain used in Mode B, with $\gamma = 2.0$, $w = 3.0$, and a predictor plus one corrector evaluation per fine step. The teacher has all hop-specific components disabled and is refreshed every 500 field updates from a field EMA with decay 0.999, maintained at each field update. The six two-step hops use the Mode B objective with an on-policy fraction of 0.5 and feedback depth drawn from 1, 2, 3 by a fixed per-update schedule. Feedback passes rebuild their input from the predicted content aggregate and outgoing estimate, with both predictions detached. The initial pass and the final additional pass, when present, are supervised against the same targets. To support longer hops at inference, three four-step hops are trained from hop update 500 onward using teacher boundary states. Their targets average the endpoints of the EMA field chain and a composition of two EMA short hops with adapters enabled, constructed from the same starting context and noise draws. The outgoing-estimate target is averaged in the same way. Concurrent training took approximately 34 hours on eight H100 GPUs, with a measured per-field-update cost of 1.24–$1.3\times$ that of plain field training. These measurements exclude the subsequent finishing stage.

Finishing and inference. After the field completes its training schedule, the hop parameters receive 5000 further updates at a constant learning rate of 3×10^{-5}, continued from the concurrent-training checkpoint and distilled from the frozen final field EMA, which adapts the hops to the field weights used at inference. At inference the model is the field EMA with these finished hop parame-

		Original sampler				Mode A (ours)				
Model	Protocol	NFE	γ	gPPL	Entropy	NFE	γ	gPPL	Entropy	NFE Reduction
ELF checkpoints										
ELF-B	SDE	32	2.0	20.24	5.09	19	2.5	20.43	5.09	**1.7**×
ELF-M	SDE	32	2.0	23.34	5.20	19	2.5	23.42	5.22	**1.7**×
ELF-L	SDE	32	2.0	28.74	5.32	19	2.5	27.45	5.33	**1.7**×
ELF-B	SDE	64	2.0	14.65	4.99	32	3.25	14.91	4.99	**2.0**×
ELF-M	SDE	64	2.0	17.95	5.13	32	3.5	18.22	5.14	**2.0**×
ELF-L	SDE	64	2.0	21.42	5.25	32	3.25	21.46	5.25	**2.0**×
LangFlow										
LangFlow	ODE	64	–	80.34	5.51	32	–	71.44	5.50	**2.0**×
LangFlow	ODE	128	–	60.09	5.43	64	–	51.42	5.41	**2.0**×
LangFlow	ODE	512	–	41.52	5.31	128	–	41.94	5.33	**4.0**×
LangFlow	ODE	1024	–	36.53	5.25	256	–	36.03	5.26	**4.0**×
LangFlow	re-noised	256	1.0	37.52	5.34	128	1.0	35.20	5.33	**2.0**×
LangFlow	re-noised	1024	0.1	33.66	5.24	256	0.5	28.67	5.23	**4.0**×
FMLM base										
FMLM base	re-noised	512	0.5	62.29	5.28	96	1.0	60.05	5.30	**5.3**×
FMLM base	re-noised	1024	0.5	55.17	5.23	128	1.0	52.79	5.24	**8.0**×

Table 3: **Mode A: training-free sampling across FLM families.** AIM-FLM matches or improves the quality–diversity operating point of the original sampler with substantially fewer network evaluations. On ELF-B/M/L, the reduction is 1.7–$2.0\times$; on LangFlow it reaches $4\times$; and on the one-hot FMLM base it reaches $8\times$. Each row compares the same released checkpoint under the corresponding sampling protocol. NFE in this table excludes the final decoder or readout. Entropy is reported in nats.

ters and guidance $w = 3.0$. The noise scale γ is selected per budget with the weights and guidance fixed, taking the lowest gPPL among settings whose entropy is at or above that of the field's default 32-step sampler; the selected values, together with the gPPL-first alternative $\gamma = 2.5$, are listed in Tab. 6. Three layouts are reported (Tab. 5): three four-step hops with two passes each followed by the standard tail (8 NFE), six two-step hops with two passes each followed by the standard tail (14 NFE), and the same six hops with the decoder-only readout (12 NFE). NFE counts network evaluations without the final decoder call; the tail and decoder evaluations are those of the field itself, with the hop parameters inactive. The reference applies ELF's official SDE sampler to the same final field EMA with the hop-specific components disabled, at its default 32 steps and at equal NFE, with γ likewise selected per budget (Tab. 6). Both generation pipelines therefore use the same field checkpoint rather than separately trained backbones. The main Mode C results include no additional last-hop adaptation; the concurrent-versus-post-hoc control is in Tab. 7.

C Additional Experiments and Details

C.1 Additional Results for Mode A

Tab. 3 evaluates the training-free sampler across ELF model scales, LangFlow, and the one-hot FMLM base. These results test whether joint state–estimate refinement transfers across architectures, representation spaces, and deterministic or re-noised sampling protocols.

C.2 Additional Results for Mode B

Tab. 4 complements the main few-step comparison with two additional regimes. The first compares Mode B with published distilled samplers at equal inference budgets. The second tests compression of substantially longer reference or teacher trajectories.

		Reference			Mode B (ours)			
Base	Reference	NFE	gPPL	Entropy	NFE	gPPL	Entropy	Result
Against published few-step samplers at equal NFE								
ELF-B	ELF+PD	1	136.10	5.26	1	89.81	5.29	**−34% gPPL**
ELF-B	ELF+PD	4	34.33	5.16	4	26.41	5.12	**−23% gPPL**
ELF-B	ELF+PD	8	23.20	5.07	8	21.79	5.11	**−6% gPPL**
FMLM base	FMLM	2	133.29	5.25	2	103.25	5.27	**−23% gPPL**
FMLM base	FMLM	4	111.31	5.26	4	83.12	5.30	**−25% gPPL**
Compression of long chains at matched quality								
ELF-B	ELF+PD, 32 steps	32	21.30	5.04	12	19.54	5.08	**2.7× fewer**
ELF-M	released SDE sampler	16	40.00	5.33	4	36.64	5.36	**4.0× fewer**
ELF-M	released SDE sampler	20	33.60	5.29	7	31.40	5.34	**2.9× fewer**
LangFlow	Euler sampler	256	49.24	5.37	16	49.74	5.36	**16× fewer**
LangFlow	Euler sampler	512	41.52	5.31	24	40.90	5.31	**21× fewer**
LangFlow	Euler sampler	1024	36.53	5.25	32	35.47	5.26	**32× fewer**
FMLM base	teacher chain	128	65.12	5.30	12	64.75	5.28	**10.7× fewer**

Table 4: **Mode B: few-step distillation and long-chain compression.** *Top:* at equal NFE, Mode B reduces gPPL by 6–34% relative to the corresponding published few-step samplers while maintaining similar entropy. *Bottom:* the same feedback-preserving hop formulation compresses substantially longer reference or teacher trajectories, using 2.7–32× fewer network evaluations at comparable quality. NFE in this table excludes the final decoder or readout where applicable. Entropy is reported in nats.

Sampler on the same field	NFE	γ	gPPL	Entropy
Official SDE sampler	8	2.0	84.87	5.23
Official SDE sampler	12	2.0	46.59	5.19
Official SDE sampler (default)	32	1.5	22.10	5.06
Mode C, 3 hops ×2, standard tail	8	2.3	21.71	5.07
Mode C, 6 hops ×2, decoder only	12	2.25	20.37	5.07
Mode C, 6 hops ×2, standard tail	14	2.2	19.66	5.07

Table 5: **Mode C: matched-field comparison with the native sampler.** AIM-FLM and the reference use the same final field EMA; the reference disables the hop-specific parameters and runs ELF's official SDE sampler. At 8 NFE, the three-hop student reaches the quality of the default 32-NFE sampler, while the 12- and 14-NFE configurations further improve gPPL at nearly unchanged entropy. Guidance is $w = 3$ and γ is selected per budget. NFE in this table excludes the final decoder call.

C.3 Additional Results for Mode C

Mode C asks whether the same few-call structure can be learned concurrently with the generative field itself. All comparisons use one final ELF-B field EMA: the reference disables the hop-specific parameters and runs ELF's official sampler, while AIM-FLM enables them. Table 5 gives the three reported layouts against the field's sampler at its default budget and at equal NFE. Table 6 lists every operating point, including the refine readout and the gPPL-first noise scale, together with the teacher chain and the training-free Mode A chain on the same field. Table 7 compares the concurrently trained hop parameters with post-hoc students trained on the frozen final field, which separates the effect of concurrent training from that of student capacity. The training and inference configuration is given in Appendix B.3.

D Qualitative visualization

Please refer to Figs. 6 and 8.

Sampler on the same field	Readout	NFE	γ	gPPL	Entropy
Official SDE sampler	decoder	8	1.5	85.48	5.30
Official SDE sampler	decoder	8	2.0	84.87	5.23
Official SDE sampler	decoder	12	1.5	50.63	5.25
Official SDE sampler	decoder	12	2.0	46.59	5.19
Official SDE sampler (default)	decoder	32	1.5	22.10	5.06
Teacher chain, 12 steps (training-time)	standard tail	26	2.0	18.66	5.01
Mode A chain, 16 fine steps	standard tail	34	2.5	18.69	5.02
Mode C with entropy-first noise scale					
Mode C, 3 hops $\times 2$	decoder	6	2.3	23.84	5.08
Mode C, 3 hops $\times 2$	refine	7	2.3	22.63	5.08
Mode C, 3 hops $\times 2$	standard tail	8	2.3	21.71	5.07
Mode C, 6 hops $\times 2$	decoder	12	2.25	20.37	5.07
Mode C, 6 hops $\times 2$	refine	13	2.3	19.72	5.06
Mode C, 6 hops $\times 2$	standard tail	14	2.2	19.66	5.07
Mode C with gPPL-first noise scale					
Mode C, 3 hops $\times 2$	decoder	6	2.5	22.31	5.05
Mode C, 3 hops $\times 2$	refine	7	2.5	21.23	5.05
Mode C, 3 hops $\times 2$	standard tail	8	2.5	20.37	5.04
Mode C, 6 hops $\times 2$	decoder	12	2.5	18.90	5.03
Mode C, 6 hops $\times 2$	refine	13	2.5	18.52	5.04
Mode C, 6 hops $\times 2$	standard tail	14	2.5	17.98	5.03

Table 6: **Mode C: complete operating points on the final field.** The upper block reports the native field sampler, the 12-step teacher chain used to generate hop targets, and the training-free Mode A chain on the same field. The middle block uses budget-specific γ values that keep entropy at or above the default field sampler. The lower block uses $\gamma = 2.5$ to prioritize gPPL. The *refine* readout applies one additional field corrector to the committed state before decoding, while the standard tail applies one SDE step and one Euler step. At 14 NFE with $\gamma = 2.5$, Mode C reaches 17.98 gPPL compared with 18.66 for the 26-NFE teacher chain, with 0.016 higher entropy. A second seed set reproduces all Mode C operating points within 1.5%. NFE excludes the final decoder call.

Student on the frozen final field	NFE	γ	gPPL	Entropy
Post-hoc, full network (105M)	4	1.75	29.71	5.09
Post-hoc, full network (105M)	4	2.0	26.91	5.05
Post-hoc, rank-64 adapters (9.8M)	4	2.0	29.68	5.07
Post-hoc, full network (105M)	8	2.0	22.38	5.03
Post-hoc, rank-64 adapters (9.8M)	8	2.0	23.42	5.04
Concurrently trained adapters (Mode C)	8	2.3	21.71	5.07
Official SDE sampler (default)	32	1.5	22.10	5.06

Table 7: **Concurrent versus post-hoc hop training on the same field.** Post-hoc students are trained after freezing the final field, using the Mode B recipe with either the full network or rank-64 adapters. At 8 NFE, increasing post-hoc student capacity from adapters to the full network closes only part of the gap, while concurrently trained adapters reach 21.71 gPPL at 5.07 entropy and recover the default field quality. This comparison indicates that the Mode C gain is not explained only by student parameter count. NFE excludes the final decoder call.

NFE = 19

AIM-FLM Mode A, ELF-B (released checkpoint)

Gen. PPL: 16.5
Entropy: 5.07

In reality, a couple would live in a perfect location that included a safe desk, a wonderful library, a wonderful restaurant, multi-colored walls, a fun theater, and an amazing sports venue. When a couple of people start to live, there are many challenges to consider. The best way to do this safely and safely is to give each a couple a chance to create a perfect design house, ensuring their house is safe, and ensuring their house is always open and stable. However, there are still often a number of scenarios where a couple without their children live in the same location. There are many couples with a number of children who choose to live together so that they can live in their preference in a safer home. A number of these happen either with a different child or who choose to live with a different parent. The couple often develop strong relationships and relationships with their parents, a relationship that will keep them happy and happy with their children. The majority of these couple only leave their family because their house is nearby, and, if they don't get chance, their parent's decision to leave will prove unhelpable. On the other hand, if you are going to focus on building your own design house, you should know how to start your own house in the right location for you. Here are some of the common lessons you will need to learn to know before you start to build your own design house: Creating a great design house One of the first things you will need to know about your own design house is creating a great house. After you start to build your own design house, it is time to dive into some of the best design-house buildings in the world and a look at some of the best design buildings in the world. The Design House is a traditional office-style building located in Queensville, Maryland. It is located on the top of the 6th floor of the former George W. Bush Office Building, and is open to private and small businesses. It has dual metal walls and patios, spanning across a wide variety of styles. The traditional office style building datess back to the early 1st century BC, and has multiple buildings and logos. Each building had hand-sculpted banners to represent a specific design. More than 30 years after the building was constructed, designer Elice Wynner wanted to create a small office building structure that would consist of a small office hall and a large desk. To provide a larger space for the building to operate, she decided to design a large, individually designed structure that would allow you to use multiple offices within each building. Alongside both the desk walls, there can be a desk and a metal container that can be placed on the bottom of the building. To allow the desk to function the way you originally designed, it must be placed on the bottom floor. It should be clean and safe, but it should be cleaned out so that it remains clean and easy to use. A desk or metal container can be used to store, or you can use a metal container to store it, or a laptop or a metal container to store a paper table to storage. Here are a few common tricks to ensure your outside desk looks good: What do you need to do? What is the key to your design? What do you need to do to design something outside? The most important thing to do is to make sure your outside desk looks like itself. Before you start to develop your design, you need to change the decor to make the outside desk look similar to that of a laptop. This will ensure that the white color on the outside doesn't dry out. You will also want to improve the look so that the true green color on the outside does look real. Let me start with the answer to the question: What are the principles that you need to use for your design? What are the your favorite principles for your house? What are the most of the basic principles that you need to use for your design? What are your favorite principles for designing your house? How Do I Design? There are a lot of different design practices for each house. There are a lot of ways to go about designing something that you really want to look like. Design is always an hard process. You have to make it look like a specific design, and you cannot get that done without constantly changing your design. There are plenty of steps that you can take to make sure your designed house stays by the normal rules of your design process. You can check out the guides below. There are many different approaches that you can use when determining a design for your house: What is what you want to see on the internet? What do your favorite design shows look like? What is your favorite TV show on the internet? What would you think of a TV show in fashion?

Figure 6: Example OpenWebText generation from AIM-FLM Mode A on the released ELF-B at 19 NFE (16 fine steps with joint refinement, $\gamma = 2.5$, $w = 3$, standard tail).

NFE = 2

AIM-FLM Mode B, ELF-B, four-hop student as two hops

Gen. PPL: 61.2
Entropy: 5.17

6. What where to go won't be an easy task. First of all, you have places to run up to the right line. There are a variety of that aren't just running. Regardless from size or terrain, you can play a game stage to run to the next route. If it's rain over or off your track, go on or off the field when it's in cold of cold or rain. You can go out to a nearby a water park or park, to walk back to the dark line, as well as go at any of the dangerousest sites on the field. Although it's hard to travel, but fortunately, it's always easy to be right off. Before your trip's game, you'll need a train driver to in a safe suit, but you can't be there. Even if you are planning for the season game and want to make up to ground, you may want some time to go to an old club, to hike into into the Mediterranean, or walk to a football park in South Lanburg. When you leave the game, the buss will run off and head off the field, and when the bus returns to the dry line, and you'll be sure they're on ready your way to the next point. And to get it done, there aren't much issues with a foreign trainer, but even if you are a boat fan, there are chance you are you won't be there. If you know, this is a great idea to where you to stay in a setting for your family or team fans. Take a buddy, I take a bus coach has all-going travel and also be as easy to learn as a travel trip, when the coach just says it's the right time. There is also plenty to be lotsa little preparation things ahead to take a great trip. While this post is said to be very popular of the author and has a lot of reviews from his clients and critics on their publications. In fact, I've drew a lot of interest in local business discussions in a town like Santa Palma Street or any other location down the Main Road. If you have enjoyed my thoughts on travel here, then here will be some places and ideas to should enjoy to explore. The Summer Season will be the first day in Madrid, Spain on September 31, and 29 in, as the first league clubs in the world have been known with' most greatest divisions. Of course they have the most successful contenders in the Champions League, but the club has been active in competition in particular in the past. If one is the best to stay to Barcelona is you can find a variety of locations and options, although you'll need to visit a Champions League Game Park, stay at the local soccer center, a private soccer practice park, and an international seafood store. In addition to other restaurants and campgrounds, it's easy to pick up your local trains, shoot them, take teats, sit in your patio area and explore the, before relaxing in the water. 10. What other players you want when you travel to Anbica In course, though, this may be a big bit to. There is one other, that can't be dismissed in your right way. In the 2012 semifinals, Bayern Munich saw a ball from the fireing on the field, deteriorating Barca's ability to secure the European league points last after before them to break away. Since the start to the season, the Champions League has been quiteterribly successful. In 2014, they were defeated as the Reds were 19th place in the Champions League Cup top group, followed by the 3-2 defeat over Manchester City in November. In 2012's defeat in final, when the Reds were angled from the league lead after United after Robert dropped away after retirement, finally dropping the German club to final, but was dis much failure. Arsenal have won their top in the season's matches since the 2017 Champions League World Championship on September 15. Madrid:: winners at Arsenal, Manchester City and London Spurs. 5. What other players you you go to Barcelona, though, and this are probably the best place to go to at Madrid? As for me, I think, Barieri's performance beauty comes from the team's style, and extraordinary quality of displayed in the back room. This list is full of the best choices for the top items, with not a guess, there are a lot details that make you're looking for the president. Austria's foreign officials, include, Manuel Trump, Manuel Mueller andhow, Trump emphasizes the role of creating a national party' is " to be hostile of each other" in Germany, and he hass to be understanding with a new foreign policy strategy..

Figure 7: Example OpenWebText generation from the AIM-FLM Mode B student on ELF-B at 2 NFE ($\gamma = 1.55$, $w = 3$, decoder-only readout).

NFE = 8

AIM-FLM Mode C, ELF-B trained from scratch with hop adapters

Gen. PPL: 18.1
Entropy: 5.05

Alas Olens played for Manchester City from 1993 to 2005. The former Manchester United striker helped lead City to the Champions League title last season and made a move for City. However, a deal was made at the start of the season. The club has confirmed that Real Arca are interested in striker Carlos En Olens. Steve O'Brien has also been working on the deal, and the club is hoping to flourish in the future. Olens representatives have not declined to comment on the deal. "I'm working closely with the City manager to see whether the deal is appropriate for the club's ongoing development process," Olens said. "I'm excited to see them join City, so they won't ask for a new deal." The Olens brothers are also members of the United States Football League (IFL), the Youth League, and the United States Football League. In 2013, En Olens played for the club against City in the English Champions League, after making 23 appearances that season. He became a partner for the club in 2013 and spent his first two seasons as City's new head coach when Olens was appointed as manager. The player has always played a central role in the club's film and television network expansion process, which includes services for newspaper and TV services, local newspapers, internet networks, and media and entertainment services, as well as local and national channels such as CBSNBC and CNN. When the weather gets hot, you aren't alone. When you are downtown, you might want to make a list of the small areas, where you want to go and where you should go. A List of Major City's Top 20 Small areas In this post, we'll be help you in choosing a area for you. Do You Need to Pick a Small Area Before you start a home? From here, we've got some path guides that help you in choosing a small area. There's also path guides that help you choose a small area for you. These path guides take you through the time it will take to start a home. First off, make sure you pick the option for the small area to help you plan into building a home. There are a wide range of options to build a home over only a few square area, or even build a home over a small area for a night. If you're looking for a job or way to build a home, you'll always be able to build one, or set up a theater for a good night. There are a wide range of options, including a theater, a music area, a theatre center, or a stage center.Whether you're looking to build a dance center, or setting up a theater area for a night, watching radio shows, or whatever, try to consider these options. It's true, building an area is a very important decision for you. Building a home is a process at the beginning of your life, and having a sized area is an important part offiguring out the details of your plans. Here are some important things you should know before you start making a home. If you don't live in small city, or live in a small city, you really will need to be aware of what your area will look like and what it is. The area is perfectly designed, the environment is clean, and the arts is the art and the environment. Many cities want to use an area as a place for business, but there are some things you'll need to know about. Make an area look like it's too big or too thin. For example, maybe a small area isn't a piece, and maybe it's a piece of wood. If it's too thick, you need to make a roof in there, to prevent the area from slipping out into the ground. When you're choosing an area, always be careful. It's important to be aware of it before making a home. If you want to look innovative, don't try to be aware of it, because the more attractive a home is, the better the design process will become. Before making your home look great, it's important to make sure it looks beautiful, and you need to be familiar with the design. Artfacts have always been a vital part of design your home, but when you're looking for something unique, don't use the art, and don't use the art. Don't use something like a piece of wood. There are some rules on why making a home is important when making your plans. Here are some rules that will help you figure out how your home looks, which you should know before you choose a home.

Figure 8: Example OpenWebText generation from AIM-FLM Mode C at 8 NFE (three four-step hops with two passes each, standard tail, $\gamma = 2.3$, $w = 3$).