DISCRETE BECKMANN TRANSPORT MODELS
FOR ONE-STEP LANGUAGE MODELING AND REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete diffusion and flow models are a promising alternative to autoregressive language models, but compressing many-step sampling into fewer steps typically requires distilling a pretrained teacher model. This caps the student at the teacher's quality and requires a costly two-stage training pipeline. We introduce **Discrete Beckmann Transport Models** (DBTM), built on a time-independent flow whose autonomous transport map provably carries any point in the ambient space to a fixed point on the vertices of the simplex in a single step. We show that this fixed-point property is characterized by a conservation equation whose residual can be minimized directly from data, removing the requirement for a teacher flow and time conditioning. Under this construction, a partially trained map corresponds to the flow truncated at finite time, so generation reduces to iterating one map until it reaches a fixed point. We further extend the map to a partial-context interpolant where additional function evaluations act as refinement steps rather than ODE integration steps. On language modeling and reasoning tasks, DBTM enables one- and few-step generation that improves quality and accuracy over discrete diffusion and continuous flow baselines.

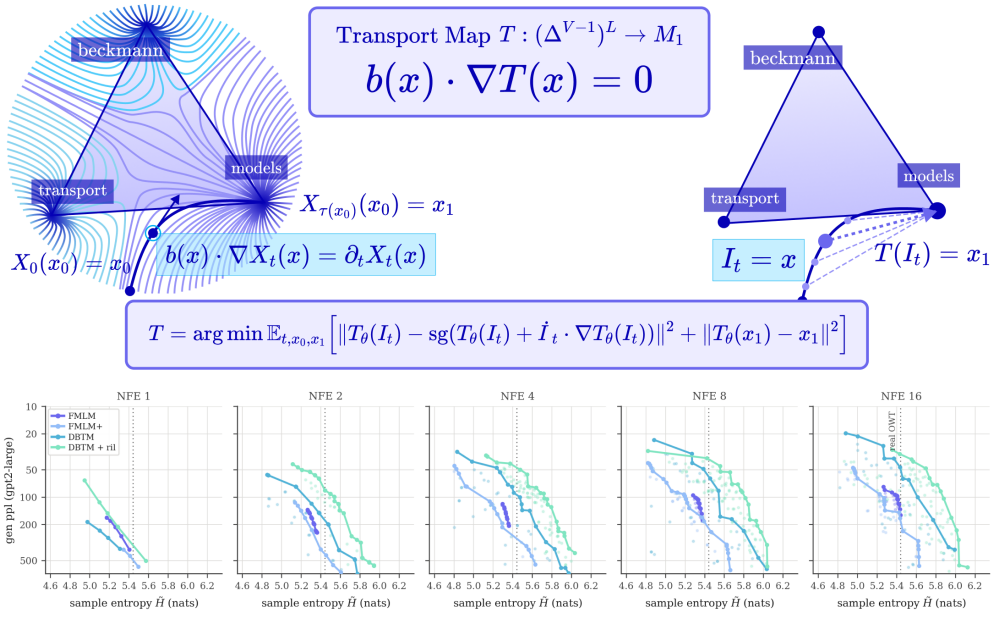

Figure 1: **Discrete Beckmann Transport Models. Top Left:** The autonomous velocity field $b(x)$ on the 2-simplex. **Top Right:** The transport map $T(x)$ solving the conservation equation $b \cdot \nabla T = 0$ with $T(x) = x$ on M_1 maps every point along the interpolant directly to its endpoint x_1. **Bottom:** Generative frontiers of DBTM vs. flow map baselines FMLM and FMLM+ on OpenWebText (lower Gen. PPL and higher entropy are better).

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.

1 INTRODUCTION

Discrete diffusion models (Shi et al., 2024; Sahoo et al., 2024; Ou et al., 2025; Zheng et al., 2025) are competitive alternatives to autoregressive language modeling, offering bidirectional dependencies. However, for sequences of length L over a vocabulary of size V, the state space grows as V^L, forcing practical samplers to factorize the per-step denoising distribution across positions. This factorization discards intra-step dependencies, so unmasking many positions at once degrades sample quality, and high-fidelity generation requires many function evaluations scaling linearly with L. The resulting limit on parallelism has motivated continuous-space flows that land on the simplex and can be distilled into few-step samplers. Among these techniques are *flow maps* (Boffi et al., 2025) which have been adapted to discrete sequences (Potaptchik et al., 2026; Lee et al., 2026a) by defining a mean denoiser that transports a point at time s directly to its further denoised state at time t, landing at a vertex of the simplex at $t = 1$.

A key limitation of few-step language modeling with flow maps is the need to learn both the diagonal ($s = t$) and off-diagonal ($s \neq t$) terms, either by distilling a teacher flow approximating the marginal velocity b_t or by training both jointly via bootstrapping, with the former commonly performing better (Potaptchik et al., 2026; Lee et al., 2026a). Distillation caps the student at the teacher's quality, propagating its approximation errors, and imposes a two-stage pipeline that must train the teacher to convergence before fitting the student. Bootstrapping removes the teacher but makes the off-diagonal targets depend on the model's own evolving predictions, which can be unstable and slow to converge.

We introduce **Discrete Beckmann Transport Models (DBTM)**, which learn the *autonomous transport map* T of a time-independent flow that provably has straighter trajectories than the time-dependent flow and carries any ambient point to the data manifold supported on the simplex vertices. This fixed-point property is formalized by the *conservation equation* $b \cdot \nabla T = 0$, whose residual we minimize to train directly from data, replacing distillation entirely. Our **main contributions** are:

1. **Discrete Beckmann Transport Models:** A *one-step map* $T(x) := X_{\tau(x_0)}(x)$ of the time-independent flow ODE that provably transports any ambient point to a fixed point on the discrete data manifold without time conditioning.
2. **Direct Training from Data:** T is trained without distillation by regressing onto a frozen copy of itself plus an Euler correction. A partially trained T_θ maps to an intermediate point along the autonomous flow, so the manifold is reached by iterating T_θ.
3. **Self-Correction via Iterative Refinement:** A partial-context interpolant enables confidence- or quality-based iterative refinement during training and inference.

On unconditional language modeling and conditional reasoning, DBTM outperforms baselines at lower or matched NFEs. Related work is discussed in App. A.

2 PRELIMINARIES

Continuous Generative Modeling on the Simplex Let $\mathcal{V} = \{1, \dots, V\}$ be a vocabulary and $x = (x^1, \dots, x^L) \in \mathcal{V}^L$ a sequence of L tokens. Embedding each token as a vertex of the probability simplex $\Delta^{V-1} := \{x \in \mathbb{R}^V_{\geq 0} : \sum_j x_j = 1\}$ via the one-hot map $x^\ell \mapsto e_{x^\ell}$ represents a sequence with a point in $(\mathbb{R}^V)^L$. The data distribution μ_1 is atomic, supported on the V^L vertex configurations $M_1 := \{e_1, \dots, e_V\}^L \subset (\Delta^{V-1})^L$, and we decode by the per-position argmax, which is exact on M_1. The prior is a product of Gaussians $\mu_0 = \mathcal{N}(0, \sigma^2 \boldsymbol{I}_V)^{\otimes L}$, and continuous generative modeling aims to transport μ_0 to μ_1 via a continuous flow in ambient space.

Generative Flows and Flow Maps The stochastic interpolants framework (Albergo and Vanden-Eijnden, 2023; Albergo et al., 2025) connects μ_0 and μ_1 through the interpolant:

$$I_t := \alpha_t x_0 + \beta_t x_1, \quad t \in [0, 1] \tag{1}$$

with $x_0 \sim \mu_0$, $x_1 \sim \mu_1$ independent and $\alpha_0 = \beta_1 = 1$, $\alpha_1 = \beta_0 = 0$. We take $\beta_t = 1 - \alpha_t$ with $\alpha_t = (1 - t)^a$ for $a \geq 1$, recovering the linear interpolant at $a = 1$. Writing $\mu_t := \mathrm{Law}(I_t)$, the flow matching velocity is learnable by regression without access to μ_t:

$$b_t(x) := \mathbb{E}_{x_0, x_1}\left[\dot{I}_t \middle| I_t = x\right] = \arg\min_{\hat{b}_t} \mathbb{E}_{x_0, x_1}\left[\left|\hat{b}_t(I_t) - \dot{I}_t\right|^2\right] \tag{2}$$

The pair (b_t, μ_t) satisfies the *continuity equation* $\nabla \cdot (b_t \mu_t) = -\partial_t \mu_t$, and the probability flow ODE $\frac{d}{dt} X_{s,t}(x) = b_t(X_{s,t}(x))$, $X_{s,s}(x) = x$, transports μ_s to μ_t at a cost measured in network function evaluations (NFE). Flow maps instead learn the two-time solution operator $X_{s,t}$, advancing the state over a finite interval in one evaluation. By uniqueness of the ODE solution, it obeys the semigroup property:

$$X_{t,u} \circ X_{s,t} = X_{s,u} \text{ for all } s \le t \le u, \qquad X_s \circ X_u = X_{s+u} \text{ when } b_t \equiv b \tag{3}$$

where the second identity is the case where the velocity field is time-independent.

3 Autonomous Flows on the Simplex

3.1 Properties of the Autonomous Flow

While standard flow matching learns a time-dependent field b_t and integrates it over a fixed horizon $t \in [0, 1]$, an autonomous flow model (Lee et al., 2026b) defines a time-independent velocity field b:

$$b(x) := \mathbb{E}_{t,x_0,x_1}[\dot{I}_t | I_t = x] = \arg\min_{\hat{b}} \mathbb{E}_{t,x_0,x_1}\left[|\hat{b}(I_t) - \dot{I}_t|^2\right], \qquad t \sim \mathcal{U}[0,1] \tag{4}$$

associated with the autonomous ODE:

$$\dot{X}_s(x_0) = b(X_s(x_0)), \quad X_0(x_0) = x_0 \tag{5}$$

where $s \in [0, \infty)$ is the integration time of the autonomous ODE, which is decoupled from the interpolation time t.

Proposition 3.1 (Time-average of standard flow matching)**.** *Given the time-dependent flow matching velocity $b_t(x) := \mathbb{E}_{x_0,x_1}[\dot{I}_t | I_t = x]$ defined with the interpolant $I_t := \alpha_t x_0 + \beta_t x_1$, the autonomous velocity field is the time-weighted average:*

$$b(x) = \frac{\int_0^1 \mu_t(x) b_t(x) dt}{\int_0^1 \mu_t(x) dt} \tag{6}$$

The proof is in App. C.1.1. Intuitively, $b(x)$ averages the velocities of every interpolant passing through x at any time $t \in [0, 1]$, so the field depends on position alone, removing the time dependence that constrains the flow to terminate at μ_1 at $t = 1$ and yielding straighter dynamics. We make this precise in App. C.1. Just as the time-dependent flow satisfies a *continuity equation*, the autonomous flow satisfies the *divergence condition* of Beckmann's continuous transportation problem (Proof in App. B.2) (Lee et al., 2026b; Beckmann, 1952) obtained by integrating over time $t \in [0, 1]$:

$$\int_0^1 \nabla \cdot (b_t \mu_t) = -\int_0^1 \partial_t p_t \implies \nabla \cdot j = \mu_0 - \mu_1 \tag{7}$$

where $j = \nu b = \int_0^1 \mu_t b_t dt$ is the *current* and $\nu(x) := \int_0^1 \mu_t(x) dt$ the *occupation measure*. This makes μ_0 the source and μ_1 the sink of the current, so b transports μ_0 to μ_1 – with the key distinction that X_s need not reach M_1 at $s = 1$. We call the time at which it converges to M_1 the *hitting time* $\tau(x_0) \in [0, +\infty)$, and show it is finite on the simplex.

Theorem 3.1 (Convergence of autonomous flow on the simplex)**.** *Let μ_0 be absolutely continuous on $(\mathbb{R}^V)^L$ with continuous, strictly positive density, and let*

$$M_1 = \mathrm{supp}(\mu_1) = (\{e_1, \dots, e_V\})^L \subset (\Delta^{V-1})^L \tag{8}$$

be the finite set of V^L vertex configurations. Then for μ_0-a.e. x_0, the autonomous flow $X_s(x_0)$ satisfies:

- *(i)* ***Absorption****: if $X_{s^\star}(x_0) \in M_1$ for some $s^\star < \infty$, then $X_s(x_0) = X_{s^\star}(x_0)$ for all $s > s^\star$.*
- *(ii)* ***Convergence****: there is $x^\star(x_0) \in M_1$ with $\lim_{s\to\infty} X_s(x_0) = x^\star(x_0)$.*
- *(iii)* ***Finite hitting time****: for the linear interpolant $I_t = (1-t)x_0 + tx_1$, $\tau(x_0) := \inf\{t \ge 0 : X_s(x_0) \in M_1\} < +\infty$. Although the autonomous velocity field vanishes at the*

vertices, $b(e_j) = 0$ for all $e_j \in M_1$, it does not vanish in the limit approaching them: $\lim_{r\downarrow 0} b(e_j + r\omega) = -\kappa(\omega)\omega$. The approach speed is bounded below by $\kappa_- > 0$ and $\tau(x_0) \le t_{entry} + 2r_1/\kappa_-$ for a fixed radius r_1.

The proof is in App. C.4. The resulting discontinuity in b at the vertices is an artifact of the linear interpolant: for a general interpolant (1) that is *self-stopping*, i.e., $\dot{\alpha}_t \to 0$ as $t \to 1$, Proposition C.5 shows b is continuous at the vertices with $b(x) \to 0$ as $x \to e_i$.

3.2 Autonomous Transport Map

Given the hitting time $\tau(x_0)$ of the autonomous flow $X_s(x_0)$, the associated one-step transport map T can be written as the limit:

$$T(x_0) := \lim_{s \to \tau(x_0)} X_s(x_0) \in M_1, \quad T_{\#}\mu_0 = \mu_1 \tag{9}$$

which is also the pushforward of μ_0 to μ_1. Since $X_s(x_0)$ is a deterministic function of x_0, starting along any point $X_s(x_0)$ for $s \in [0, \tau(x_0))$ yields the same fixed point $T(X_s(x_0)) = T(x_0)$. By the semigroup condition (3), the map T is idempotent such that $T^{\circ k}(\cdot) = T^{\circ k'}(\cdot)$ for any $k, k' \in \mathbb{N}$.

Proposition 3.2 (Autonomous transport map solves conservation equation (Lee et al., 2026b))**.** *Given the autonomous flow velocity field b, the autonomous flow $X_s(x_0)$ defined in (5) solves the Eulerian equation:*

$$\begin{cases} b(x) \cdot \nabla X_s(x) = \partial_s X_s(x), \quad X_0(x) = x & x \notin M_1 \\ X_s(x) = x & x \in M_1 \end{cases} \tag{10}$$

Since the autonomous transport map $T(x)$ defined in (9) is the limit of $X_s(x)$ as $s \to \tau(x)$ with vanishing velocity at the vertices ($\dot{X}_s(e_j) = 0$ for all $j \in \{1, \dots, V\}$) by Theorem 3.1, $T(x)$ is the unique solution of the conservation equation:

$$\begin{cases} b(x) \cdot \nabla T(x) = 0 & x \notin M_1 \\ T(x) = x & x \in M_1 \end{cases} \tag{11}$$

Proposition 3.2 follows from Theorem 3.1, with proof in App. C.6. This is the key result that drives a principled approach for training the one-step map directly from data without distillation.

4 Learning One-Step Maps Without Distillation

We present a method of directly learning the one-step transport map T by matching the updates that converge to the solution of the conservation equation in (11). The converged map T transports any noisy state to its corresponding fixed point on the data manifold, yielding one-step generation and principled refinement techniques.

4.1 Iteratively Solving the Eulerian Equation

Unlike previous few-step model training, which requires distillation or bootstrapping from an approximated target, Proposition 3.2 presents a principled approach to learning T by iteratively regressing the autonomous flow $X_s(x)$ until the equilibrium point when $b(x) \cdot \nabla X_{\tau(x)}(x) = b(x) \cdot \nabla T(x) = 0$ for all $x \in (\mathbb{R}^V)^L$. Starting from an untrained map $T^{(0)}(x) = X^{(0)}(x) = x$, we iteratively regress the Euler step of size η:

$$X^{(n+1)}(x) = X^{(n)}(x) + \eta b(x) \cdot \nabla X^{(n)}(x) \tag{12}$$

where T_θ is trained to match $X^{(n+1)}$ given fixed $X^{(n)}$ as $X^{(n)} \to T$ for large n. Since we define $b(x) := \mathbb{E}_{t,x_0,x_1}[\dot{I}_t | I_t = x]$, the optimal map is:

$$T = \arg\min_{T_\theta} \mathbb{E}_{t,x_0,x_1} \left[\| T_\theta(I_t) - \mathrm{sg}(T_\theta(I_t) + \dot{I}_t \cdot \nabla T_\theta(I_t)) \|^2 + \| T_\theta(x_1) - x_1 \|^2 \right] \tag{13}$$

minimized exactly when T satisfies the conservation equation $\mathbb{E}_{t,x_0,x_1}[\dot{I}_t \cdot \nabla T_\theta(I_t) | I_t = x] = b(x) \cdot \nabla T_\theta(x) = 0$ for $x \notin M_1$ and $T(x) = x$ for $x \in M_1$. Crucially, this objective ensures that even

when T_θ has not converged, it approximates a map to some point along X_s, so by the semigroup property (3), k inference-time iterations $T_\theta^{\circ k}(x_0)$ approximates along the flow to $X_{kt}(x_0)$, which converges to M_1 as $kt \to \tau(x_0)$.

4.2 Training Objectives for the Transport Map

In practice, the autonomous transport map is parameterized as $T_\theta(\cdot) = \mathrm{softmax}(f_\theta(\cdot))$ for a neural network f_θ. Since the map is restricted to the simplex, we formulate each objective as a cross-entropy (CE) over categorical distributions rather than an unconstrained squared error, with the exception of the transport loss where the target does not lie on the simplex.

Transport Loss From (13), the *transport loss* is obtained by regressing T_θ onto the stop-gradient of itself, added to the Euler step correction along the autonomous flow:

$$\mathcal{L}_{\text{transport}}(\theta) := \mathbb{E}_{t,x_0,x_1}\left[\sum_{\ell\in[L]\setminus\mathcal{C}} \left\| T_\theta(I_t)_\ell - \mathrm{sg}(T_\theta(I_t) + \dot{I}_t \cdot \nabla T_\theta(I_t))_\ell \right\|^2\right] \tag{14}$$

where $\mathcal{C}$ is the set of committed tokens and the correction $\frac{d}{dt}T(I_t) = \dot{I}_t \cdot \nabla T(I_t)$ is tangent to the simplex and its elements sum to zero.

Semigroup Loss From (9), we show that by the semigroup condition $T_\theta^{\circ k}(\cdot) = T_\theta^{\circ k'}(\cdot)$, which we enforce using the *semigroup loss*:

$$\mathcal{L}_{\text{semi}}(\theta) := \mathbb{E}_{t,x_0,x_1}\left[\sum_{k\in\mathcal{K}}\sum_{\ell\in[L]\setminus\mathcal{C}} \mathrm{CE}\left(T_\theta(I_t)_\ell, \mathrm{sg}(T_\theta^{\circ k}(I_t)_\ell)\right)\right] \tag{15}$$

for $k \in \mathcal{K} \subset \mathbb{N}$ applications of the map. We apply this loss only toward the end of training, since the initial identity map $T = \mathrm{id}$ satisfies it trivially.

Boundary Loss To enforce the boundary condition $T(x) = x$ for $x \in M_1$ from Proposition 3.2, we add a *boundary loss*:

$$\mathcal{L}_{\text{bnd}}(\theta) := \mathbb{E}_{x_1}\left[\sum_{\ell\in[L]\setminus\mathcal{C}} \mathrm{CE}\left(T_\theta(x_1)_\ell, x_1^\ell\right)\right] \tag{16}$$

Beyond the boundary itself, any intermediate I_t should map to its target, $T_\theta(I_t) = x_1$. While the *anchor condition* is implicitly enforced by the transport loss, the geometry of the simplex makes explicit enforcement useful over a specific time interval, which we describe next.

4.3 Phase Transitions and the Anchor Loss

The flow constructed from the time-dependent interpolant on the simplex commits to a vertex at a short time window called the *phase transition*. Before the transition, the output token remains uncertain; after it, the token is committed, and further integration does not change the output. Such transitions have been analyzed in diffusion models (Ambrogioni, 2025; Yu and Huang, 2025), and we can use tools from random energy models (REM) (Derrida, 1980) to determine the timing of this transition.

Theorem 4.1 (Phase Transition and Commitment Time)**.** *The phase transition where the interpolant (1) commits to a vertex occurs at time:*

$$t^\star = 1 - \left(1 + \sigma\sqrt{2\log V}\right)^{-1/a} \tag{17}$$

where σ is the noise scale of the x_0, and a is the exponent on $\alpha_t = (1-t)^a$ and V is the vocabulary size defining the simplex Δ^{V-1}.

The full derivation is in App. C.2. The derivation relies on the idea that competing forces pull the flow trajectory towards each of the V vertices, resembling the selection of states in a random energy model at a given temperature, which undergoes a phase transition at $t = t^\star$. With the target vertex fixed at e_k during training, the trajectory starts from x_0, where each vertex carries an energy proportional to the noise scale σ, and the flow commits to the target when the energy of the target vertex outweighs the

competing energies of all other vertices $j \neq k$. Intuitively, this means that the interval $[0, t^\star]$ is when the flow must resolve uncertainty in coordination with the other positions, and the interval $[t^\star, 1]$ is the period post-transition where the endpoint is decided, and we can safely anchor the trajectory to the dominant vertex.

This gives a principled criterion for *when* to supervise the map toward its target. Over $[t^\star, 1]$, where the endpoint is already decided, we can anchor $T_\theta(I_t)$ directly without sacrificing diversity:

$$\mathcal{L}_{\text{anchor}}(\theta) := \mathbb{E}_{t,x_0,x_1}\left[\sum_\ell \mathbf{1}[t > t_{\text{anchor}}]\text{CE}\left(T_\theta(I_t)^\ell, x_1^\ell\right)\right] \tag{18}$$

with $t_{\text{anchor}} = t^\star$. Empirically, disabling $\mathcal{L}_{\text{anchor}}$ or setting $t_{\text{anchor}} > t^\star$ slowed convergence, while $t_{\text{anchor}} < t^\star$ caused mode collapse (Table 12). More broadly, this characterization of the commitment time on the simplex suggests that the transport loss (14) and inference-time steering are best concentrated on the pre-transition interval $[0, t^\star]$, where the endpoint is still being resolved.

5 Training and Sampling with Fixed-Point Refinement

5.1 Test-Time Scaling with Self-Refinement

Partial-Context Interpolant We consider the subclass of interpolants in which tokens at positions $\mathcal{C} \subset \{1, \ldots, L\}$ in the sequence are held as *context tokens* in their clean state $I_t^\ell = x_1^\ell$, while the remaining are noised to time t:

$$I_t^\ell(\mathcal{C}) := \begin{cases} x_1^\ell & \ell \in \mathcal{C} \\ \alpha_t^\ell x_0^\ell + (1 - \alpha_t^\ell) x_1^\ell & \ell \in [L] \setminus \mathcal{C} \end{cases} \tag{19}$$

The map T_θ remains time-independent but is trained on this larger family of per-position interpolants $I_t^\ell(\mathcal{C})$, enabling conditioning on partial clean contexts during inference.

Confidence-Based Refinement Using the predicted map T_θ, we can obtain a measure of the model's per-token confidence q_ϕ^ℓ and use it to determine which tokens to *renoise* and *refine* via the context-dependent map. Concretely, we define:

$$q^\ell(T_\theta(x)) := \langle T_\theta(x), \hat{x}_1 \rangle, \quad \hat{x}_1 = \arg\max(T_\theta(x)) \tag{20}$$

Given a threshold κ (e.g. $\kappa = 0.9$), the refinement step holds the clean one-hot tokens $\hat{x}^\ell$ for the set of high-confidence positions $\mathcal{C} := \{\ell : q^\ell(T_\theta(x)) \geq \kappa\}$ and samples fresh noise for all remaining positions $\ell \in [L] \setminus \mathcal{C}$, and reapplies the map T_θ, which training with the partial-context interpolant makes well-defined.

Refinement with Learned Per-Token Quality Another approach to performing self-correction on the output of the transport map is to measure the coherence of each token given the predicted context and resampling noise for incoherent predictions. The ideal coherence score is the probability of the proposed token $\hat{x}_1^\ell$ given the ground truth at all other positions, $q_\star^\ell(\hat{x}) := \Pr[\hat{x}^\ell \mid x_1 \oplus m^\ell]$, where m^ℓ masks the ℓth position. Since x_1 is unavailable at inference, computing it would cost L extra forward passes. Following Kim et al. (2026), we replace the ground truth with the model's own prediction and define the *per-token quality*:

$$q_\phi^\ell(T_\theta(x)) = \Pr\left[\hat{x}_1^\ell = x_1^\ell \mid T_\theta(x)\right], \quad \hat{x}^\ell = \arg\max\left(T_\theta(x)^\ell\right) \tag{21}$$

which can be estimated by a learnable quality head with parameters ϕ trained by minimizing:

$$\mathcal{L}_\phi(\phi) = \mathbb{E}_{x_0,x_1}\left[\sum_\ell \text{BCE}\left(\mathbf{1}[\hat{x}_1^\ell = x_1^\ell], q_\phi^\ell(T_\theta(x))\right)\right], \quad \hat{x}_1 = \arg\max\left(T_\theta(x)\right) \tag{22}$$

Since this expectation is over the same (x_0, x_1) pairs as the map objectives, it trains jointly with T_θ. At inference, obtaining q_ϕ costs no additional NFE and defines $\mathcal{C} := \{\ell : q_\phi^\ell(T_\theta(x)) \geq \kappa\}$ just like confidence-based refinement. Algorithm 2 provides the full inference algorithm with refinement.

5.2 Training with Refinement in the Loop

At inference, $\mathcal{C}$ is not random as in training but built up over rounds by committing the highest-scoring positions first. The training distribution therefore contains out-of-distribution contexts, since none of the losses enforce the map to complete the structured context generated from the inference sampler. The mismatch is significant in the few-NFE regime: under a budget of k rounds, each round commits roughly $1/k$ of the remaining positions, so the context after round r is a large, highly non-uniform set, and q_ϕ is also out of distribution when scoring it.

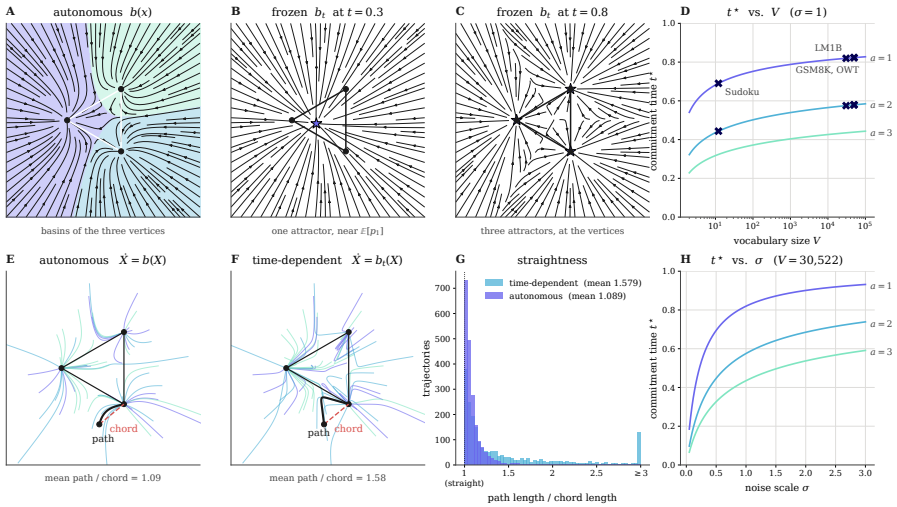

Figure 2: **Dynamics of Autonomous and Time-Dependent Flows. (A)** The autonomous flow on Δ^2 is stationary and flows toward fixed basins at the vertices. **(B)** The time-dependent flow before the commitment time ($t = 0.3$) has its basin at the mean, and **(C)** after it ($t = 0.8$) at the vertices. **(E), (G)** Mean path and chord ratio of autonomous vs. time-dependent flows. **(D), (H)** Commitment time $t^\star$ vs. simplex dimension V and prior scale σ for $a \in \{1, 2, 3\}$.

Commit Rule We close this gap by supervising T_θ on the commit sets the sampler is likely to visit at inference. Index refinement rounds by $r = 1, \ldots, k$ with $\mathcal{C}_0 = \emptyset$, and let $\mathcal{R}_r := [L] \setminus \mathcal{C}_{r-1}$ be the positions still uncommitted entering round r. Writing q^ℓ for the commit score of position ℓ (confidence or the quality q_ϕ^ℓ), each round commits the set $\Delta\mathcal{C}_r$ ranked highest by q^ℓ and renoises the rest:

$$\Delta\mathcal{C}_r := \underbrace{\{\ell \in \mathcal{R}_r : q^\ell \geq \kappa\}}_{\text{threshold}} \cup \underbrace{\operatorname{top}_{n_r}\left(\mathcal{R}_r; q\right)}_{\text{floor}}, \qquad n_r := \left\lceil \tfrac{|\mathcal{R}_r|}{k-r+1} \right\rceil, \qquad \hat{x}^\ell_{r,k} := \begin{cases} x_1^\ell & \ell \in \mathcal{C}_r \\ x_0^\ell & \ell \in [L] \setminus \mathcal{C}_r \end{cases} \tag{23}$$

where $\operatorname{top}_{n_r}(\mathcal{R}_r; q)$ denotes the n_r highest-scoring positions of $\mathcal{R}_r$, and $\mathcal{C}_r := \mathcal{C}_{r-1} \cup \Delta\mathcal{C}_r$. So every position above κ is committed, and at minimum n_r positions are committed regardless. Since $|\Delta\mathcal{C}_r| \geq n_r$ and $n_k = |\mathcal{R}_k|$, we have $\mathcal{C}_k = [L]$ and the sampler always terminates within k NFE.

Refinement-in-Loop Objective The states $\hat{x}_{r,k} := \textsc{Renoise}(T_\theta(\hat{x}_{r-1,k}))$ act as partial-context interpolants whose context comes from the sampler itself. We train the map to recover x_1 from them via the *refinement-in-loop* (ril) objective over uncommitted positions (committed ones are pinned one-hot and carry no gradient):

$$\mathcal{L}_{\text{ril-c}}(\theta) := \mathbb{E}_{x_0, x_1}\left[\sum_{k \in \mathcal{K}} \sum_{r=1}^{k} \sum_{\ell \in [L] \setminus \mathcal{C}_r} \operatorname{CE}\left(T_\theta(\hat{x}_{r-1,k})^\ell,\ x_1^\ell\right)\right] \tag{24}$$

The same states train the quality head via $\mathcal{L}_{\text{ril-}\phi}$ (163), and late in training a self-distillation variant $\mathcal{L}_{\text{semi-r}}$ matches every intermediate round to the final k-round output, removing the need for a prescribed noise-data coupling (App. D). The full objective is

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{transport}} + \lambda_{\text{semi}}\mathcal{L}_{\text{semi}} + \lambda_{\text{bnd}}\mathcal{L}_{\text{bnd}} + \lambda_{\text{anchor}}\mathcal{L}_{\text{anchor}} + \lambda_\phi\mathcal{L}_\phi + \lambda_{\text{ril}}\left(\mathcal{L}_{\text{ril-c}} + \lambda_{\text{quality}}\mathcal{L}_{\text{ril-}\phi}\right) \tag{25}$$

The stop-gradient rollout costs k extra forward passes and no extra backward pass. Since q_ϕ is a head on the trunk of T_θ, a refinement round costs exactly one NFE, so the k-round sampler has *no inference overhead*. Alternative instantiations are given in App. D.

6 Experiments

We show that (i) the autonomous flow has stable attractors at the simplex vertices and smoother trajectories; (ii) anchoring at the phase transition prevents mode collapse and speeds convergence; (iii) direct training of the one-step map scales to language modeling and reasoning; and (iv) refinement-in-loop training improves coherence and accuracy.

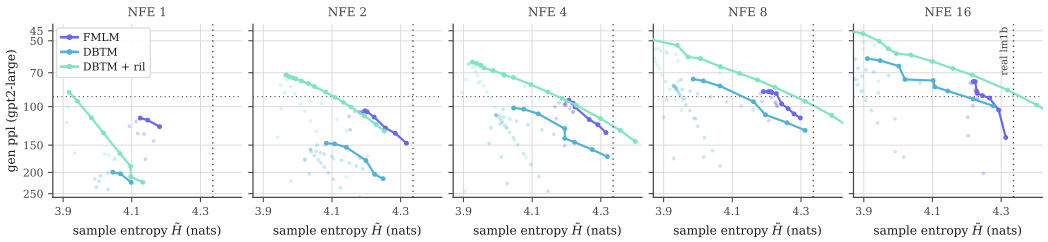

Figure 3: **Generative Frontiers of DBTM vs. FMLM.** Gen. PPL (log scale, ↓) against entropy (↑) on LM1B for NFE $\in \{1, 2, 4, 8, 16\}$. Each curve is the Pareto frontier over all inference-time knobs swept for that method (DBTM: commit temperature, prior scale σ, threshold κ; FMLM: churn γ, prior scale). The dotted line marks the LM1B data entropy (4.336). DBTM and DBTM + ril are trained for 200K steps; FMLM uses a 1M-step teacher and 100K distillation steps. Full sweep details in Table 13.

	LM1B						OWT					
NFE	1		2		4		1		2		4	
Method	Gen-PPL ↓	Ent. ↑	Gen-PPL ↓	Ent. ↑	Gen-PPL ↓	Ent. ↑	Gen-PPL ↓	Ent. ↑	Gen-PPL ↓	Ent. ↑	Gen-PPL ↓	Ent. ↑
Duo + DCD	1224.52	4.33	520.08	4.20	210.88	4.23	5743.29	6.02	891.16	5.41	250.86	5.37
Duo + Di4C	292.94	3.79	247.69	3.87	150.67	4.00	370.51	3.92	210.22	4.63	154.67	4.85
MDLM + SDTT	1429.48	4.31	602.14	4.28	241.01	4.28	1260.86	5.26	877.22	5.34	339.73	5.38
MDLM + Di4C	1217.10	4.38	621.59	4.37	247.32	4.00	1298.80	5.29	758.23	5.35	239.27	5.40
CFM	269.72	3.10	267.39	3.15	267.97	3.28	–	–	–	–	–	–
FMLM	119.34	4.16	110.19	4.21	98.76	4.21	168.30	5.17	133.29	5.25	111.31	5.26
FMLM+	–	–	–	–	–	–	378.53	5.34	114.46	5.14	41.16	4.77
DFM (PSD)	94.08	4.06	87.42	4.08	78.89	4.10	180.29	4.91	152.83	5.03	122.32	5.10
DFM (ESD)	68.11	3.79	77.60	4.11	71.53	4.13	5.33	0.26	108.91	5.15	77.08	5.27
DBTM	201.20	4.03	167.05	4.19	119.99	4.20	188.2	4.97	76.2	5.14	52.2	5.26
DBTM + ril	85.50	3.92	81.00	4.05	73.54	4.04	65.2	4.95	62.2	5.38	57.5	5.52

Table 1: **DBTM Unconditional Language Modeling.** Gen. PPL (↓) and entropy (↑) across NFEs on LM1B and OWT. OWT values for DBTM use linear attention at matched parameter count; the standard DiT comparison and the full NFE sweep are in Table 7, and matched-entropy frontier comparisons in Tables 13 and 14.

6.1 The Autonomous Field Has Stable Attractors at the Vertices

Experimental setup. We compare the time-dependent flow matching field against the autonomous field on the 2-simplex Δ^2, visualizing both fields and comparing the mean path and chord ratio of their trajectories (Figure 2; details in App. C.1.2).

The autonomous flow is straighter. The autonomous field partitions the ambient space into basins whose only equilibria are the vertices, which we prove for general simplices in App. C.1. Its stationary basins yield straighter trajectories (Figure 2E-G) than the time-dependent flow, whose basins appear at staggered times in the simplex interior before moving to the vertices.

6.2 Scaling the One-Step Map to Language

Experimental setup. We evaluate unconditional generation on LM1B (Chelba et al., 2013) and OpenWebText (OWT) (Gokaslan et al., 2019), reporting generative perplexity (Gen-PPL) under GPT-2-Large (Radford et al., 2019) and within-sequence unigram entropy (Ent.). Since a model collapsing onto repeated tokens can reach arbitrarily low perplexity (Pynadath et al., 2026b), we compare perplexities only among checkpoints whose entropy is in the range of the data entropy. Baselines are distilled discrete diffusion (Duo and MDLM distilled with DCD, Di4C, or SDTT), continuous flow matching (CFM), and flow maps (FMLM, FMLM+, DFM), evaluated at matched NFE. We report **DBTM**, trained with the transport, boundary, and anchor losses (Section 4.2), and **DBTM + ril**, which adds the refinement-in-loop objectives (Section 5.2). Details in App. G.2.

Discrete BTM is competitive with distilled flow maps. On LM1B (Table 1), DBTM + ril outperforms every distilled discrete diffusion model at 1, 2, and 4 NFE and is competitive with the best flow-map baseline while keeping entropy close to the data. On OWT, DBTM + ril achieves the lowest perplexity among all non-collapsed methods at every NFE with the closest entropy to the data. At 4 NFE, DBTM reaches the lowest perplexity overall (52.2) while DBTM + ril attains the highest entropy (5.52), tracing the two ends of the quality-diversity frontier. Refinement-in-loop reduces

1-NFE perplexity by 58% on LM1B and 65% on OWT. On the perplexity-entropy frontier (Figures 1, 3), both variants dominate the baselines across data-range entropies, with no teacher, fewer training steps, and no time conditioning. Samples are in App. I.

Anchoring at the phase transition prevents mode collapse. Theorem 4.1 gives $t^\star$ as a function of the vocabulary size, noise scale, and schedule exponent. Sweeping t_{anchor} around this value on LM1B (Table 12) demonstrates its importance. Anchoring before $t^\star$ supervises toward a fixed target while the endpoint is undecided, and the model collapses to repetitive tokens, whereas anchoring after $t^\star$ (e.g. $t_{\text{anchor}} = 0.85$) leaves the post-transition interval unsupervised, resulting in higher Gen. PPL at equal training time (Table 12).

Refinement enables more effective use of step budget. Evaluating a single DBTM checkpoint across NFE budgets (Table 7), quality improves consistently with budget until the sequence is fully committed, where it saturates. Distilled flow models instead move along a fixed time axis in discrete jumps, and masked diffusion unmasks on a fixed schedule with factorization error growing as NFE shrinks. One model produces a one-shot proposal and then, via Section 5, renoises low-quality positions and remaps them against committed context. The commitment condition also acts as a self-stopping rule, so the NFE count adapts to the sample rather than being fixed in advance as in flow map and flow matching samplers.

6.3 Reasoning with Discrete BTMs

Experimental setup and baselines. We evaluate Sudoku at three difficulty levels by one-shot exact-solve accuracy, and TinyGSM code (Liu et al., 2023a) by zero-shot accuracy on the GSM8K test set (Cobbe et al., 2021) following Kim et al. (2025). Baselines are many-step autoregressive, discrete diffusion (MDLM, Duo), and continuous diffusion (CANDI (Pynadath et al., 2026a), FLM, S-FLM (Lee et al., 2026a)) models, and FMLM+ (Agarwal et al., 2026), a distilled few-step flow map trained for iterative refinement, to which we match NFE budget and parameter count. Details in App. G.

		Sudoku				GSM8K	
Category	Method	NFE	Easy	Med.	Hard	NFE	Acc. (%,↑)
AR	Sample	128	13.9	5.1	0.6	512	53.9
	Greedy	128	14.6	5.1	1.0	512	63.3
Discrete	MDLM	128	92.0	77.1	30.2	1024	18.0
	Duo	128	96.3	84.7	58.4	1024	17.2
Continuous	CANDI	128	79.3	45.9	16.7	1024	0.2
	FLM	128	94.2	82.7	44.5	1024	0.3
	S-FLM	128	94.8	85.2	45.0	1024	18.0
Few-Step	FMLM+	4	97.9	92.0	71.2	1	0
	FMLM+	16	97.8	92.6	81.4	32	16.6
	DBTM + ril	4	**99.5**	**97.3**	**84.6**	1	**0.8**
	DBTM + ril	16	**99.9**	**99.4**	**97.5**	32	**16.8**

Table 2: **Discrete Beckmann Transport Model Reasoning Performance.** Sudoku columns are exact-solve accuracy (%) on the 2000-puzzle held-out set and GSM8K is accuracy (%) on the 1319-problem test set. FMLM+ at NFE 16 is our own evaluation of the released checkpoints on easy and medium, and the hard and GSM8K cells are taken from Table 13 of Agarwal et al. (2026).

Refinement enables fixed-point reasoning. Refinement matters most for reasoning, since a one-shot decode of a puzzle or arithmetic chain almost always contains locally plausible but globally inconsistent positions. On Sudoku (Table 2), DBTM matches or exceeds all many-step baselines at every difficulty with one to two orders of magnitude fewer NFEs, and exceeds FMLM+ at the same budget, with the margin widening on harder puzzles. On GSM8K, DBTM reaches 0.8% in 1 NFE, surpassing CANDI and FLM at 1024 NFEs and FMLM+, and 16.8% at 32 NFEs, slightly above FMLM+ and closing the gap with S-FLM at 32× fewer NFE. AR methods still dominate GSM8K, suggesting refinement that exploits left-to-right structure as future work. Further results are in App. D.

7 Conclusion

We introduced **Discrete Beckmann Transport Models** (DBTM), which replace the time-dependent velocity of flow matching with a stationary, autonomous field on the simplex whose straighter, time-averaged trajectories have their only stable equilibria at the vertices. Defining the transport map as the solution to the conservation equation along this field lets us train it end-to-end from data, with no time conditioning and no teacher. Since every map application proposes a clean sequence, additional NFEs become renoise-and-refine steps that self-correct given committed context rather than finer integration steps. Across language modeling and reasoning benchmarks, DBTMs improve on both discrete diffusion and flow baselines.

AI USE STATEMENT

Generative AI tools were used for manuscript language editing and for assistance with code drafting and debugging. All reported numerical results were produced by the authors' computational pipelines and checked against saved outputs. The authors reviewed and revised all AI-assisted text and code and take responsibility for the final manuscript, analyses, and artifacts.

REPRODUCIBILITY STATEMENT

We will make the code publicly available on GitHub and provide installation scripts with documentation to address the dependency issues. Furthermore, all the datasets used in this paper are publicly available, with the experimental details provided in the Appendix. We hope that this will support and accelerate future research and development.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. We propose a general method for few-step generation of discrete sequences, evaluated on unconditional language modeling (LM1B, OpenWebText) and reasoning (Sudoku, TinyGSM/GSM8K). All datasets are publicly available, and no human subjects were involved. Our models are small-scale research models, but, like other language models, models trained on web text may reproduce biases or harmful content present in their training data, and faster sampling could lower the cost of generating such content at scale. We encourage standard safeguards such as data filtering and output moderation when applying these methods.

REFERENCES

Manan Agarwal, Sheel Shah, Chanhyuk Lee, Jaehoon Yoo, Jerry Huang, Seunghoon Hong, Aditi Raghunathan, Jinwoo Kim, and Nicholas M Boffi. 2026. Posterior refinement: Fast language generation via any-order flow maps. *arXiv preprint arXiv:2606.24773*.

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. 2025. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80.

Michael S Albergo and Eric Vanden-Eijnden. 2023. Building normalizing flows with stochastic interpolants. *International Conference on Learning Representations*.

Luca Ambrogioni. 2025. The statistical thermodynamics of generative diffusion models: Phase transitions, symmetry breaking and critical instability. *Entropy*.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*.

Iskander Azangulov, Kianoosh Ashouritaklimi, Leo Zhang, Simon Vary, and Patrick Rebeschini. 2026. Masked language flow models. *arXiv preprint arXiv:2606.27617*.

Martin Beckmann. 1952. A continuous model of transportation. *Econometrica: Journal of the Econometric Society*, pages 643–660.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. 2024. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions in Machine Learning Research*.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. 2025. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. 2024. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *International Conference on Machine Learning*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

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

Chaoran Cheng, Jiahan Li, Jiajun Fan, and Ge Liu. 2025. α-flow: A unified framework for continuous-state discrete flow matching models. *arXiv preprint arXiv:2504.10283*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İ Ceylan, Michael Bronstein, and Avishek J Bose. 2024. Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information Processing Systems*.

Bernard Derrida. 1980. Random-energy model: Limit of a family of disordered models. *Physical Review Letters*, 45(2):79.

Justin Deschenaux and Caglar Gulcehre. 2025. Beyond autoregression: Fast llms via self-distillation through time. In *International Conference on Learning Representations*.

Justin Deschenaux and Caglar Gulcehre. 2026. Language modeling with hyperspherical flows. *arXiv preprint arXiv:2605.11125*.

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

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. 2025. One step diffusion via shortcut models. *International Conference on Learning Representations*.

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

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

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

Yi Guo, Wei Wang, Zhihang Yuan, Rong Cao, Kuan Chen, Zhengyang Chen, Yuanyuan Huo, Yang Zhang, Yuping Wang, Shouda Liu, et al. 2025. Splitmeanflow: Interval splitting consistency in few-step generative modeling. *arXiv preprint arXiv:2507.16884*.

Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. 2025. Distillation of discrete diffusion through dimensional correlations. *International Conference on Machine Learning*.

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

Jaehyeong Jo and Sung Ju Hwang. 2025. Continuous diffusion model for language modeling. *Advances in Neural Information Processing Systems*.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Koo, Nam Ik Cho, et al. 2026. Parallelbench: Understanding the trade-offs of parallel decoding in diffusion llms. In *International Conference on Learning Representations*, volume 2026.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. 2026. Fine-tuning masked diffusion for provable self-correction. *International Conference on Machine Learning*.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. 2025. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *International Conference on Machine Learning*.

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

Cheuk-Kit Lee, Florentin Coeurdoux, Yuyuan Chen, Sophia Tang, Peter Potaptchik, Yilun Du, Michael S Albergo, and Eric Vanden-Eijnden. 2026b. Beckmann transport models: From autonomous flows to one-step maps. *arXiv preprint arXiv:2608.01692*.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2023. Flow matching for generative modeling. *International Conference on Learning Representations*.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. 2023a. Tinygsm: achieving> 80% on gsm8k with small language models. *arXiv preprint arXiv:2312.09241*.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2023b. Flow straight and fast: Learning to generate and transfer data with rectified flow. *International Conference on Learning Representations*.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. *International Conference on Machine Learning*.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2025. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *International Conference on Learning Representations*.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182. IEEE.

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

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. 2026a. Candi: Hybrid discrete-continuous diffusion models. *International Conference on Machine Learning*.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. 2026b. Generative frontiers: Why evaluation matters for diffusion language models. *arXiv preprint arXiv:2604.02718*.

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

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, Ismail Ilkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. 2026. Categorical flow maps. *International Conference on Machine Learning*.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. 2025. Align your flow: Scaling continuous-time flow map distillation. *Advances in Neural Information Processing Systems*.

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

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. 2025. The diffusion duality. *International Conference on Machine Learning*.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in Neural Information Processing Systems*.

Yang Song and Prafulla Dhariwal. 2024. Improved techniques for training consistency models. *International Conference on Learning Representations*.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency models. *International Conference on Machine Learning*.

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. 2024. Dirichlet flow matching with applications to dna sequence design. *International Conference on Machine Learning*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Sophia Tang and Pranam Chatterjee. 2026. Expanding flow maps. *arXiv preprint arXiv:2607.21585*.

Sophia Tang, Yinuo Zhang, Alexander Tong, and Pranam Chatterjee. 2025. Gumbel-softmax flow matching with straight-through guidance for controllable biological sequence generation. *arXiv preprint arXiv:2503.17361*.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. 2024. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions in Machine Learning Research*.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. 2025. Gated delta networks: Improving mamba2 with delta rule. *International Conference on Learning Representations*.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2024. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*.

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

Zhendong Yu and Haiping Huang. 2025. Nonequilbrium physics of generative diffusion models. *Physical Review E*, 111(1):014111.

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

Linqi Zhou, Mathias Parger, Ayaan Haque, and Jiaming Song. 2026. Terminal velocity matching. *International Conference on Learning Representations*.

A Related Work **15**
A.1 Discrete and Continuous Generative Models for Sequence Data 15
A.2 Time-dependent and autonomous flows 15
A.3 Flow maps in discrete state space 15

B Background on Autonomous Flows **16**
B.1 Notation 16
B.2 Divergence Condition 16

C Theoretical Results **18**
C.1 Dynamics of Time-Dependent and Autonomous Flows 18
C.2 Deriving the Phase Transition and Commitment Time 26
C.3 Occupation Measure of Autonomous Flow on the Simplex 29
C.4 Autonomous Velocity Vanishes at the Vertices 31
C.5 Convergence of Autonomous Flow on the Simplex 32
C.6 Eulerian and Conservation Equations 35

D Refinement-in-Loop Details and Additional Results **37**
D.1 On-policy Rollouts 37
D.2 Refinement-in-Loop for Reasoning 38
D.3 Refinement-in-Loop for Unconditional Language Modeling 39

E Hyperparameter Discussion **41**
E.1 Training Hyperparameters 41
E.2 Inference Hyperparameters 41

F Ablations and Additional Results **42**

G Experiment Details **47**
G.1 General Experiment Details 47
G.2 Unconditional Language Modeling Experiment Details 47
G.3 Reasoning Experiment Details 48

H Algorithms **50**

I Example Generations **52**

A Related Work

A.1 Discrete and Continuous Generative Models for Sequence Data

Generative modeling of discrete data in the form of sequences or graphs has taken many forms. Discrete diffusion (Austin et al., 2021; Lou et al., 2024), masked discrete diffusion (Shi et al., 2024; Sahoo et al., 2024; Ou et al., 2025; Zheng et al., 2025), and discrete flow matching (Gat et al., 2024; Campbell et al., 2024; Azangulov et al., 2026) model discrete sequence generation as a continuous-time Markov chain (CTMC) where transitions are defined by a rate matrix. However, since the discrete state space grows exponentially with the vocabulary size and sequence length, discrete diffusion models rely on learning the factorized approximations of the transition probabilities that require many inference steps to capture the dependencies between tokens for accurate generation (Kang et al., 2026). To overcome the factorization errors inherent in discrete state-space parameterizations, continuous state-space parameterizations have emerged as an alternative by embedding the flow trajectories in simplex space (Stark et al., 2024; Davis et al., 2024; Tang et al., 2025) or in a latent embedding space (Cheng et al., 2025; Chen et al., 2026; Dieleman et al., 2022; Hu et al., 2026; Deschenaux and Gulcehre, 2026; Yang et al., 2026; Jo and Hwang, 2025). In this work, we model the autonomous flow transporting a Gaussian prior μ_0 on $\mathbb{R}^d$, $d := VL$, to the data distribution μ_1 supported on the vertex set $\mathcal{V} := \{e_1, \dots, e_V\}^L$ of the product of simplices $(\Delta^{V-1})^L \subset \mathbb{R}^d$, which is the construction used in discrete flow maps (Potaptchik et al., 2026; Lee et al., 2026a).

A.2 Time-dependent and autonomous flows

Our framework develops the discrete state space instantiation of *Beckmann Transport Models (BTMs)* (Lee et al., 2026b), named after Beckmann's continuous model of transportation (Beckmann, 1952), an approach to the problem of transporting states between distributions μ_0 and μ_1 via a time-independent velocity field b that satisfies the divergence condition $\nabla \cdot (\nu b) = \mu_0 - \mu_1$. In contrast to the time-dependent velocity field of standard flow matching (Lipman et al., 2023; Albergo and Vanden-Eijnden, 2023; Albergo et al., 2025; Liu et al., 2023b) that transports x along $b_s(x) = \mathbb{E}_{x_0,x_1}[\dot{I}_s | I_s = x]$ which is the expectation of the interpolant velocity passing x at time s, the autonomous velocity field additionally averages over the time coordinate of the interpolant $s \in [0, 1]$, yielding the expectation over all interpolant velocities that pass x at any time. In our work, we derive and empirically validate the unique properties of the autonomous flow on the simplex, specifically showing that it yields straighter flows with favorable Lipschitz properties and stationary attractors at the vertex. We establish the fact that the autonomous velocity field is non-vanishing approaching each vertex but vanishes when reaching it, partitioning the ambient space around the simplex into stationary basins of attraction.

A.3 Flow maps in discrete state space

There exists a long line of work on distilling and constructing few-step generative models that aim to reduce the number of function evaluations needed to integrate the generative ODE or SDE. These include consistency models (Song et al., 2023; Song and Dhariwal, 2024), shortcut models (Frans et al., 2025), MeanFlow (Guo et al., 2025), rectified flow (Liu et al., 2023b), flow maps (Boffi et al., 2024; Sabour et al., 2025; Boffi et al., 2025), and many of these frameworks have been extended to the discrete state space (Roos et al., 2026; Lee et al., 2026a; Potaptchik et al., 2026) and variable-length generation (Tang and Chatterjee, 2026). Most related to our work are flow map language models (Lee et al., 2026a) with posterior refinement (Agarwal et al., 2026) and discrete flow maps (Potaptchik et al., 2026), which learn a two-time map (s, t) along the interpolant between a Gaussian prior and the simplex vertices. Despite this shared goal, DBTM differs from flow maps in several aspects, summarized in Table 3.

Flow map models are explicitly time-conditioned and are typically trained by distilling a pretrained teacher flow b_t, requiring supervision over all (s, t) time pairs together with a careful balance of the diagonal (teacher matching) and off-diagonal consistency objectives. DBTM instead trains a single autonomous model conditioned only on the current state, with no teacher and no time embedding, reducing the space of functions that need to be learned during training. For a flow map, each NFE is a discretized jump along the generative trajectory, whereas for DBTM each NFE refines a complete clean proposal, so intermediate iterates remain in the data space and can be treated as a reasoning trace. Since inference only ever visits a small subset of the (s, t) grid, flow map models spend training time on time pairs that are never used at sampling time. Training DBTM from noise-data interpolants instead covers exactly the states encountered at inference. Finally, the autonomous formulation

admits a natural stopping criterion: sampling iterations stop once the proposal reaches its fixed point or once all tokens pass a confidence threshold, whereas flow map sampling requires the step schedule to be fixed in advance.

Table 3: **Comparison of DBTM with Flow Map Language Models (FMLM) and Discrete Flow Maps (DFM).** Both generate in a fixed small number of function evaluations, but flow maps discretize an ODE between trained (s, t) pairs and need a teacher and time conditioning, whereas DBTM refines a clean proposal with a single autonomous map. The rows contrast the two methods.

	Flow Map LMs / DFM	**DBTM (Ours)**
Few-step generation	Yes	Yes
Time conditioning	Yes (s and t conditioning)	No
Teacher model	Yes	No
Each NFE is	Discretized step along ODE	Refinement of clean proposal
Self-stopping	No	Yes
Training complexity	High; requires diagonal/off-diagonal balancing and supervision over all (s, t) time pairs	Low; trains a single model conditioned only on the current state
Training-inference mismatch	Yes; many (s, t) time jumps are unused at inference	No; training from noise-data interpolants covers all states seen at inference
Self-correction / reasoning trace	No	Yes

B Background on Autonomous Flows

B.1 Notation

Sequences and the simplex V is the vocabulary size, L the sequence length, $\ell \in [L]$ a position and j, k vertex indices. e_j is a one-hot vertex of the simplex Δ^{V-1}, $(\mathbb{R}^V)^L$ the ambient space, x^ℓ the ℓ-th position of a sequence, and M_1 the data manifold of V^L vertex configurations. μ_0 is the Gaussian prior with noise scale σ and μ_1 the data distribution.

Interpolants and time s, t, u denote time coordinates, $I_t = \alpha_t x_0 + \beta_t x_1$ the interpolant between $x_0 \sim \mu_0$ and $x_1 \sim \mu_1$, $\dot{I}_t$ its velocity and μ_t its marginal. $\alpha_t = (1-t)^a$ is the interpolant time schedule with exponent a. For the partial-context interpolant, t_ℓ and α_t^ℓ are per-position, $\mathcal{C} \subset [L]$ is the clean context set, f the clean fraction, and c_ℓ the context indicator.

Flows and the transport map b_t is the time-dependent flow matching velocity and b the autonomous (time-independent) velocity. ν is the occupation measure, $j = \nu b$ the current, $\nu^{(j)}$, $J^{(j)}$ their per-vertex components and w_j the occupancy weights. X_s the autonomous flow, $\tau(x_0)$ the hitting time of M_1 and T the autonomous transport map, the long-time limit of X_s.

Learned objects and refinement $T_\theta = \mathrm{softmax}(f_\theta)$ is the learned map, q_ϕ^ℓ the per-token quality head on the trunk hidden state h^ℓ, and $\mathrm{sg}(\cdot)$ the stop-gradient. Losses are $\mathcal{L}_{\text{transport}}, \mathcal{L}_{\text{bnd}}, \mathcal{L}_{\text{anchor}}, \mathcal{L}_\phi, \mathcal{L}_{\text{ril}}, \mathcal{L}_{\text{ril-}\phi}$ each weighted by a λ scalar. At inference, $\hat{x}_{r,k}$ is the partially committed sequence at refinement round r of k total rounds, κ is the commit threshold, $\mathcal{C}_r$ is the committed index set after round r, and NFE is the number of network evaluations, which coincides with the number of refinement rounds k.

B.2 Divergence Condition

Proposition B.1 (Autonomous flow satisfies divergence condition)**.** *For uniformly sampled $t \sim \mathcal{U}([0,1])$, the autonomous velocity $b(x) := \mathbb{E}_{t,x_0,x_1}[\dot{I}_t | I_t = x]$ satisfies the divergence condition:*

$$\nabla \cdot (b\nu) = \mu_0 - \mu_1 \tag{26}$$

Proof. For a family of interpolants $(\alpha_t, \beta_t)_{t\in[0,1]}$, the time-dependent velocity, by the chain rule $\dot{I}_t = \dot{\alpha}_t x_0 + \dot{\beta}_t x_1$, is:

$$b_t(x) = \mathbb{E}[\dot{I}_t | I_t = x] = \dot{\alpha}_t \mathbb{E}[x_0 | I_t = x] + \dot{\beta}_t \mathbb{E}[x_1 | I_t = x] = \dot{\alpha}_t \eta_0(t,x) + \dot{\beta}_t \eta_1(t,x) \tag{27}$$

where we define $\eta_0(t,x) := \mathbb{E}[x_0 | I_t = x]$ and $\eta_1(t,x) := \mathbb{E}[x_1 | I_t = x]$. b_t satisfies the continuity equation with time-dependent marginal μ_t:

$$\partial_t \mu_t + \nabla \cdot (b_t \mu_t) = 0 \tag{28}$$

The autonomous flow is given by integrating over $t \in [0,1]$ of the time-dependent flow:

$$b(x) = \mathbb{E}_t[b_t(x) | I_t = x] = \frac{\int_0^1 b_t(x)\mu_t(x)\pi(t)dt}{\int_0^1 \mu_t(x)\pi(t)dt} = \frac{1}{\nu(x)} \int_0^1 b_t(x)\mu_t(x)dt \tag{29}$$

where $\pi(t)$ is the distribution from which t is sampled (typically uniform on $[0,1]$). For uniform $\pi(t)$, we can integrate the continuity equation over $t \in [0,1]$ to get:

$$\int_0^1 \partial_t \mu_t dt + \nabla \cdot \int_0^1 b_t \mu_t dt = 0, \tag{30}$$

The first integral reduces to $\int_0^1 \partial_t \mu_t dt = \mu_1 - \mu_0$ and the second integral can be simplified using the definition of $b(x)$ in (29) to get $(\mu_1 - \mu_0) + \nabla \cdot (b\nu) = 0$ or $\nabla \cdot (b\nu) = \mu_0 - \mu_1$. □

Corollary B.1 (Endpoint parameterization)**.** *Let $q_x(t) := \mu_t(x)/\nu(x)$ be the probability density of $t \in [0,1]$ given x. Then, the endpoint parameterization of the autonomous flow is given by:*

$$b(x) = \mathbb{E}_{t\sim q_x(t)} \left[\lambda_t(\eta_1(t,x) - x)\right] = Z(x) \left(\bar{\eta}_1(t,x) - x\right) \tag{31}$$

where $Z(x) := \mathbb{E}_{t\sim q_x}[\lambda_t]$ is a scalar and $\bar{\eta}_1(t,x) := \frac{1}{Z(x)} \mathbb{E}_{t\sim q_x}[\lambda_t \eta_1(t,x)]$ is the averaged endpoint prediction.

Proof. Given (27), we can write $x = \alpha_t \eta_0(t,x) + \beta_t \eta_1(t,x)$ and for $t < 1$, we can substitute $\eta_0 = (x - \beta_t \eta_1)/\alpha_t$ to get the endpoint form:

$$b_t(x) = \frac{\dot{\alpha}_t}{\alpha_t} x + \left(\dot{\beta}_t - \frac{\dot{\alpha}_t \beta_t}{\alpha_t}\right) \eta_1(t,x) \tag{32}$$

Since $\beta_t = 1 - \alpha_t$, we have $\dot{\beta}_t = -\dot{\alpha}_t$ and the bracket simplifies to $-\dot{\alpha}\left(1 + \frac{1-\alpha_t}{\alpha_t}\right) = -\dot{\alpha}_t/\alpha_t =: \lambda_t$. Therefore, (32) reduces to:

$$b_t(x) = \lambda_t(\eta_1(t,x) - x), \quad \lambda_t := -\frac{\dot{\alpha}_t}{\alpha_t} = \frac{a(1-t)^{a-1}}{(1-t)^a} = \frac{a}{1-t} \tag{33}$$

Substituting into (29) and writing $\nu(x) = \int_0^1 \mu_t(x)dt$, we get:

$$b(x) = \frac{1}{\nu(x)} \int_0^1 \lambda_t(\eta_1(t,x) - x)\mu_t(x)dt = \int_0^1 \lambda_t(\eta_1(t,x) - x) q_x(t)dt \tag{34}$$

which is the first equality in (31). Since q_x integrates to one, $Z(x) < \infty$ and splitting the integrand linearly gives $b(x) = Z(x)\bar{\eta}_1(t,x) - Z(x)x = Z(x)(\bar{\eta}_1(t,x) - x)$, which is the second equality. □

Remark B.1 (Using the transport map to integrate the autonomous flow)**.** *Since $\eta_1(t,x)$ is a conditional expectation of $x_1 \in M_1$, $\bar{\eta}_1(t,x)$ is a convex combination of these and lies on the simplex $\bar{\eta}_1(t,x) \in (\Delta^{V-1})^L$ for every x. Therefore, the autonomous velocity is fully determined by a map onto the product of simplices, which is exactly what $T_\theta := \mathrm{softmax}(f_\theta)$ predicts. The trajectory of the autonomous flow is unchanged with respect to $Z(x)$ up to a time reparameterization, so we only need to learn the map.*

C THEORETICAL RESULTS

Here, we present the proofs and derivations for the theoretical results of our method. In App C.1, we show the favorable properties of the time-dependent and autonomous flows, including the stationary attractors at the vertices. In App C.2, we derive the commitment time of the stochastic interpolant to a vertex of the simplex as a function of the prior variance, the simplex dimension, and the exponent in the interpolant schedule. In App C.3, we derive the form of the occupation measure defining the time spent at each point in ambient space. Finally, we show in App C.4 that the autonomous flow velocity vanishes at the vertices.

C.1 DYNAMICS OF TIME-DEPENDENT AND AUTONOMOUS FLOWS

C.1.1 AUTONOMOUS FLOW AS TIME-AVERAGED DYNAMICS

Proposition 3.1 (Time-average of standard flow matching)**.** *Given the time-dependent flow matching velocity $b_t(x) := \mathbb{E}_{x_0,x_1}[\dot{I}_t | I_t = x]$ defined with the interpolant $I_t := \alpha_t x_0 + \beta_t x_1$, the autonomous velocity field is the time-weighted average:*

$$b(x) = \frac{\int_0^1 \mu_t(x) b_t(x) dt}{\int_0^1 \mu_t(x) dt} \tag{6}$$

Proof. Let $u \sim \mathrm{Unif}[0,1]$ be independent of (x_0, x_1), so that (u, I_u) has joint density $(t,x) \mapsto \mu_t(x)$ on $[0,1] \times \mathbb{R}^d$. Marginalizing in t, the density of I_u is $\bar{\mu}(x) := \int_0^1 \mu_s(x) ds$, and by Bayes' rule the conditional density of u given $I_u = x$ is $\mu_t(x)/\bar{\mu}(x)$ wherever $\bar{\mu}(x) > 0$. By independence, $\mathbb{E}[\dot{I}_u \mid u = t,\, I_u = x] = b_t(x)$, so the tower property gives:

$$b(x) := \mathbb{E}_{u,x_0,x_1}[\dot{I}_u \mid I_u = x] = \mathbb{E}\big[\, b_u(x) \,\big|\, I_u = x \big] = \int_0^1 \frac{\mu_t(x)}{\bar{\mu}(x)} b_t(x) dt \tag{35}$$

which is (6). □

C.1.2 ATTRACTORS

Proposition C.1 (Attractors of Autonomous Flow)**.** *In the dynamical system*

$$\frac{d}{dt} X_s = b(X_s), \tag{36}$$

the only attractors are the vertices of simplex $e_1, \ldots, e_V$ at all time $s \in [0, \infty)$.

Proof. The appendix C.4 already proves that velocities vanish at the vertices $e_1, \ldots, e_V$. By Proposition 3.1, given any initial point x_0, the trajectory X_s with $X_{s=0} = x_0$ approaches one of the vertices as $s \to \infty$. Therefore, the basin of attraction of the vertices $e_1, \ldots, e_V$ partitions the space $\mathbb{R}^V$. Any additional attractor comes with a non-zero measure of basin of attraction, so there are no additional attractors. □

Now we study the dynamics of time-dependent flow matching

$$\frac{d}{dt} X_t = b_t(X_t). \tag{37}$$

We study the attractors of $b_t(x_t)$ for fixed time t. As t varies, the attractors of $b_t(x_t)$ move. The trajectory of (37) is pulled by attractors that move with time. We study $b_t(X_t)$ making two assumptions below:

Assumption C.1.

(i) The prior is Gaussian $x_0 \sim \mathcal{N}(c, \sigma^2 \boldsymbol{I}_V)$ centered at $c \in \mathbb{R}^V$ with variance σ^2 and the prior is symmetric with $c_i = c_j$ for all i and j;

(ii) The interpolant $I_t = (1-t)x_0 + tx_1$ is linear.

Note that Assumption C.1 (ii) is without loss of generality, because any straight line interpolant $I_t = \alpha_t x_0 + \beta_t x_1$ is a time reparameterization of the linear interpolant $I_t = (1-t)x_0 + tx_1$. Our results hold for any linear interpolant up to a time reparameterization. The following lemma says that regardless of where the prior is centered, the attractors of $b_t(x_t)$ lie on the simplex Δ^{V-1} for any t.

Lemma C.1 (Attractors Lie on Simplex)**.** *For Gaussian prior $x_0 \sim \mathcal{N}(c, \sigma^2 \boldsymbol{I}_V)$ and any interpolant $I_t = \alpha_t x_0 + \beta_t x_1$ and any time t, all attractors lie on the simplex Δ^{V-1}.*

Proof. We state without proof that the velocity $b_t(x_t)$ under Assumption C.1 is

$$b_t(x) = \frac{1}{1-t}(D_t(x) - x), \tag{38}$$

where the denoiser $D_t(x)$ takes the form

$$D_t(x) = \sum_i w_i(x,t) e_i, \quad \text{with} \quad w_i(x,t) = \mathrm{softmax}_i\left(\log p_i + \frac{t x_i}{\sigma^2(1-t)^2} - \frac{t c_i}{\sigma^2(1-t)}\right), \tag{39}$$

where c_i is the i-th coordinates of the center c of the prior. For fixed t, an attractor $x = (x_i)_{i=1,\dots,V}$ satisfies the equation $b_t(x) = 0$, that is

$$x_i = \mathrm{softmax}_i\left(\log p_i + \frac{t x_i}{\sigma^2(1-t)^2} - \frac{t c_i}{\sigma^2(1-t)}\right), \tag{40}$$

for $i = 1, \dots, V$. Since $x = (x_i)$ takes the form of a softmax, x lie on the simplex Δ^{V-1}. Notice that $c_i = c_j$ by Assumption C.1 (i). The softmax equation (40) becomes c-independent

$$x_i = \mathrm{softmax}_i\left(\log p_i + \frac{t x_i}{\sigma^2(1-t)^2}\right). \tag{41}$$

□

Lemma C.2 (Asymptotic Formulae for Attractor Locations)**.**

(i) For small $t \gtrsim 0$, the coordinates of the unique attractor A^* are

$$x_k = p_k + \frac{p_k\,(p_k - m_2)}{\sigma^2} t + \mathcal{O}(t^2), \tag{42}$$

for $k = 1, \dots, V$, and $m_2 = \sum_j p_j^2$ is the second moment of the target distribution.

(ii) For large $t \lesssim 1$, there are V attractors labeled by $j = 1, \dots, V$. Let $\beta = \frac{t}{\sigma^2(1-t)^2}$. The coordinates of the attractors A_j are respectively given by expansions in large β

$$x_k = \delta_{kj} + c_{kj}\, e^{-\beta} + \mathcal{O}(\beta\, e^{-2\beta}), \tag{43}$$

for δ_{kj} the Kronecker delta and the coefficients c_{jk} are given by

$$c_{kj} = \begin{cases} \frac{p_k}{p_j}, & k \neq j \\ \frac{p_j - 1}{p_j}, & k = j \end{cases} \tag{44}$$

Proof. (i) is shown by setting $x_k = p_k + u_k\, t + \mathcal{O}(t^2)$ and plug into equation (40) to solve for coefficients u_k.

For (ii), let $u = 1 - t$ and set $x_k = \delta_{jk} + r_k$. Taking the ratio of the k-th and j-th equation of (41) gives

$$\frac{x_k}{x_j} = \frac{p_k}{p_j} \exp\left(\frac{t(x_k - x_j)}{\sigma^2(1-t)^2}\right). \tag{45}$$

Since $x_j \to 1$ and $x_k \to 0$ for $k \neq j$ as $t \to 1$,

$$x_k = \frac{p_k}{p_j} \exp\left(-\frac{t}{\sigma^2(1-t)^2}\right)(1 + o(1)), \qquad k \neq j. \tag{46}$$

The expression for x_j then follows from $\sum_k x_k = 1$. □

Remark C.1. *The implication of Lemma C.2 (i) is that, as t increases from 0, the unique attractor A^* moves towards the vertex l with the largest target probability, $p_l = \arg\max_k p_k$. Since the first-order coefficient $p_k(p_k - m_2) \sim p_k^2$ at $k = l$ is much larger than that at $k \neq l$. If a vertex k has $p_k < p_l^2$, and hence $p_k - m_2 < 0$, then the attractor will move away from vertex k due to the negative first-order coefficient.*

Lemma C.2 (ii) establishes the existence of an attractor near each vertex at time $t \lesssim 1$. We are interested in when the attractor A_j associated with vertex j appears. In Proposition C.2, we show that the attractor A_l corresponding to the vertex with the largest target probability $p_l = \arg\max_k p_k$ forms at $t = 0$. We call A_l the *dominant attractor*. Attractor A_j where $j \neq l$ forms much later. In Proposition C.3 and Corollary C.1, we show that the time t_j at which attractor A_j forms is given in terms of the ratio between the target probability on the j-th vertex and the highest target probability p_l.

Proposition C.2 (The Dominant Attractor)**.** *The attractor A_l corresponding to the vertex with the largest target probability $p_l = \arg\max_k p_k$ forms at time $t = 0$.*

Proof. We show that there is an attractor A^* that exists for all time t, and that attractor A^* moves to vertex l, the vertex with the largest target probability.

First, we notice that off-simplex directions are attracted to the vertex. The vector field $b_t(x)$ restricting the simplex $x \in \Delta^{V-1}$ has the same attractors as the vector field $x \in \mathbb{R}^V$. The off-simplex direction is the vector $\mathbf{1}$ of all 1s. Project the trajectory (37) onto $\mathbf{1}$, we have

$$\frac{d}{dt}\langle X_t, \mathbf{1}\rangle = \frac{1}{1-t}(\langle D_t(X_t), \mathbf{1}\rangle - \langle X_t, \mathbf{1}\rangle). \tag{47}$$

We rewrite scalar $s = \langle X_t, \mathbf{1}\rangle$, and recall that $\langle D_t(X_t), \mathbf{1}\rangle = 1$. The entries of $D_t(X_t)$ sum to 1 according to equation (39). Equation (47) becomes an equation of the projection s

$$\frac{ds}{dt} = \frac{1-s}{1-t}. \tag{48}$$

whose solution approaches $s = 1$ exponentially as t increases. Since projections onto the off-simplex direction approach 1 exponentially, the off-simplex direction attracts. Hence the vector field $b_t(x)$ restricting to the simplex $x \in \Delta^{V-1}$ has the same attractors as the unrestricted vector field for $x \in \mathbb{R}^V$.

We focus on $b_t(x)$ for x on the simplex for fixed t. Let $x = (x_i)_{i=1,\dots,V}$ and $p = (p_i)_{i=1,\dots,V}$ both be distributions. Let $\beta_1 = \frac{t}{\sigma^2(1-t)^2}$ and consider the potential function

$$V(x) = \mathrm{KL}(x \,\|\, p) - \frac{\beta_1}{2}\|x\|^2. \tag{49}$$

We show that $V(x)$ is a Lyapunov function. Denoting P_i as the i-th softmax in (39) and Z is the shared denominator among P_i for $i = 1, \dots, V$, the gradient of $V(x)$ against the i coordinate is

$$\nabla_i V(x) = -\beta_1 x_i + \log x_i - \log p_i + 1 = -\log\frac{P_i}{x_i} - \log Z + 1. \tag{50}$$

The change of potential along $b_t(x)$ is

$$\begin{aligned} b_t(x) \cdot \nabla V(x) &= \frac{1}{1-t}\sum_i (P_i - x_i)(-\log\frac{P_i}{x_i} - \log Z + 1) \\ &= -\frac{1}{1-t}\sum_i (P_i - x_i)\log\frac{P_i}{x_i}, \end{aligned} \tag{51}$$

where we have moved the constants $-\log Z + 1$ out of the sum and applied $\sum_i P_i = \sum x_i = 1$. Rewriting the above equation, we have

$$b_t(x) \cdot \nabla V(x) = -\frac{1}{1-t}(\mathrm{KL}(P \,\|\, x) + \mathrm{KL}(x \,\|\, P)) \leq 0, \tag{52}$$

where $P = (P_i)_{i=1,\dots,V}$. The equality in (52) is obtained precisely when $x = P$. That is, x is an attractor as defined in equation (40). Let x^* be the position of such an attractor. Define the Voronoi cell $C = \{x \,|\, x \in \Delta^{V-1},\ x_l > x_i, \text{for all } i \neq l\}$ as the set of points on the simplex whose closest point is the vertex l. It is clear that the minimizer x^* of $V(x)$ is in cell C; otherwise, permuting the coordinates of x^* yields a strictly lower value $V(x)$. Additionally, the minimizer of V is unique by equation (50), where $\nabla_i V(x) = 0$ has a unique solution for generic values of p_i.

We have shown there is a unique attractor A^* corresponding to the minimum of $V(x)$, and A^* is closest to the vertex l. Now we show that as $t \to 1$, the attractor A^* approaches vertex l. As $t \to 1$, we have $\beta_1 \to \infty$, and the quadratic term in $V(x)$ dominates. Under the constraint $\sum_i x_i = 1$, the quadratic term is minimized by x^* with $x^*_j = 1$ for some j and $x^*_i = 0$ for $i \neq j$. But since $x^* \in C$, it must be the vertex l with $x^*_l = 1$. □

Proposition C.3. (Times when Other Vertex Attractors Form) *Let p_l be the largest target probability and m be the number of probabilities comparable to but less than p_l. For every $j \neq l$, the time t_j when vertex attractor A_j appear is given by*

$$\frac{t_j}{(1-t_j)^2} \simeq \sigma^2 \beta^*, \quad \textit{where} \quad \beta^* - \log \beta^* = \log \frac{p_l}{p_j} + 1 + \log(m+1) \tag{53}$$

Furthermore, one can write t_j in terms of the Lambert W-function:

$$\frac{t_j}{(1-t_j)^2} = -\sigma^2 \, W\left(-\frac{p_j}{p_l(1+m)e}\right), \tag{54}$$

where e is the base of the natural logarithm.

Proof. Attractor A_j with coordinates $x = (x_i)_{i=1,\dots,V}$ satisfies the attractor equation (40). Let $\beta_1 = \frac{t}{\sigma^2(1-t)^2}$ and $\beta_2 = \frac{t}{\sigma^2(1-t)}$ denote the t-dependence in the equation. Taking the ratio between the i-th coordinate equation and the j-th coordinate equation, the normalization of the softmax cancels, and we have:

$$\frac{x_i}{x_j} = \frac{p_i}{p_j} \exp(\beta_1(x_i - x_j) - \beta_2(c_i - c_j)) = \frac{p_i}{p_j} \exp(\beta_1(x_i - x_j)). \tag{55}$$

For fixed j, change variables from p_i, x_i and c_i to h_i, ξ_i and θ_i by including the scale of β_1 into the variables.

$$h_i = \log \frac{p_i}{p_j}; \quad x_i = \frac{\xi_i}{\beta_1}; \tag{56}$$

To normalize x_i, we set

$$S = \sum_{i \neq j} \xi_i; \tag{57}$$

and the normalization condition is

$$x_j = 1 - \frac{S}{\beta_1} \tag{58}$$

Denote $\delta = \frac{S}{\beta_1}$. δ is small as attractor A_j is near vertex j by Lemma C.2 (ii). Rewriting equation (55) in the new variables and constants, we have

$$\frac{\xi_i/\beta_1}{1-\delta} = \exp(h_i + \xi_i - \beta_1 + S). \tag{59}$$

We wish to expand the left-hand side of (59) to leading order in δ. The expansion gives

$$\frac{\xi_i}{\beta_1}(1 + \mathcal{O}(\delta)) = \exp(h_i + \xi_i - \beta_i + S). \tag{60}$$

Dropping the terms higher order in δ and rearranging (60) we have

$$\xi_i \, e^{-\xi_i} = \beta_1 \, e^{h_i - \beta_1 + S}. \tag{61}$$

The attractor A_j exists when the equation above has a solution ξ_i and S satisfying the consistency equation (57). Let the solution to (61) be $\xi_i(S)$ and define

$$G(S) = \sum_{i \neq j} \xi_i(S). \tag{62}$$

Attractor A_j correspond to $\xi_i(S^*)$ and S^* satisfying the consistency equation

$$G(S^*) = S^*. \tag{63}$$

S^* is a fixed point of the map G. We know the derivative of the map G. Take the log of both sides of equation (61) and differentiate with respect to S; we have the derivatives

$$\frac{d\xi_i}{dS} = \frac{\xi_i}{1 - \xi_i}; \quad \frac{dG}{dS} = \sum_{i \neq j} \frac{\xi_i}{1 - \xi_i}. \tag{64}$$

Notice that $G(0) > 0$ as a sum of positive terms for finite β_1 and β_2. Therefore S^* is the first S such that $G(S) - S = 0$, and $G(S) - S$ must cross 0 from above, and the derivative of $G(S) - S$ at S^* is negative:

$$G'(S^*) - 1 < 0. \tag{65}$$

We stress that for the dominant attractor A_l, we have $G(0) = 0$ and $S^* = 0$, and the derivative condition (65) does not apply.

To solve equation (61) given the right-hand side, we notice that the left-hand side function $\xi_i \to \xi_i \, e^{-\xi_i}$ has two branches. For a given right-hand side, there are two solutions ξ_i, and we are interested in the solution $\xi_i \in [0, 1]$. First we notice that $\xi_i \to 0$ as $t \to 1$ because

$$\xi_i = \beta_1 x_i = \beta_1 x_j \, e^{h_i} \, e^{\beta_1 (x_i - x_j)} \leq \beta_1 \, e^{h_i} \, e^{-\beta_1 (1 + o(1))} \to 0, \tag{66}$$

where we have used that x_j and x_i tending to 1 and 0 respectively as $t \to 1$. As t and β_1 decreases, solutions $\xi_i(S^*)$ and S^* to equation (61) and (63) vary continuously. We argue that the solutions $\xi_i(S^*) < 1$. If not, consider the first time t^* such that some $\xi_i(S^*)$ approaches 1, and the rest of $\xi_k(S^*)$ are below 1. Then we have

$$\frac{\xi_i(S^*)}{1 - \xi_i(S^*)} \to \infty; \quad 0 < \frac{\xi_k(S^*)}{1 - \xi_k(S^*)} < \infty, \tag{67}$$

for $k \neq i$. And the sum of the expressions (64) is infinity:

$$G'(S^*) = \sum_{i \neq j} \frac{\xi_i}{1 - \xi_i} \to \infty, \tag{68}$$

as $t \to t^*$, contradicting (65) where the derivative $G'(S^*)$ must be less than 1. We have shown that the rescaled coordinates of A_j have $\xi_i(S^*) \in (0, 1)$.

Considering equation (61) for the largest target probability p_l while taking the log of (61), we have

$$\beta_1 - \log \beta_1 = h_l + \xi_l - \log \xi_l + S. \tag{69}$$

Recall that p_l is the largest target probability. Subtracting the l-th equation with the i-th equation, we eliminate β_1 and obtain

$$\xi_i - \log \xi_i = \xi_l - \log \xi_l + (h_l - h_i). \tag{70}$$

The map $\xi_i \to \xi_i - \log \xi_i$ is rapidly decreasing on $\xi_i \in (0, 1)$. Since $h_l - h_i = \log p_l / p_i$, if the $p_l \gg p_i$, then $\xi_l \gg \xi_i$. The sum S in (57) is dominated by m terms ξ_s, where each term corresponds to a probability p_s that is comparable to the largest probability p_l.

At time t_j, the vertex A_j disappears when the equation $G(S) - S = 0$ is tangent to 0 at $S = S^*$. For $t > t_j$, the curve $G(S) - S$ crosses 0 transversally at $S = S^*$. The tangent condition is

$$G'(S^*) = 1, \tag{71}$$

where $G'(S^*)$ is summing $\frac{\xi_i}{1-\xi_i}$ from (64). Since $\xi_l \gg \xi_i$, we have $G'(S^*)$ is dominated by m terms ξ_s, each corresponds to a probability p_s that is comparable to the largest probability p_l. We may assume all $\xi_s = \xi_l = \xi$ for some ξ for all s because of equation (70) and $h_l - h_i = \log p_l/p_s \simeq 0$. Derivative $G'(S^*)$ equaling 1 becomes

$$m\,\frac{\xi}{1-\xi} = 1 \quad \Longleftrightarrow \quad \xi = \frac{1}{1+m}, \tag{72}$$

Plugging the value of $\xi_l = \xi$ back to equation (69), we have the critical β_1^* when t_j disappears is when

$$\beta_1^* - \log \beta_1^* = h_l + \frac{1}{1+m} + \log(1+m) + \frac{m}{1+m} = h_l + 1 + \log(1+m), \tag{73}$$

where we can solve for β_1^* and hence t_j with the Lambert W-function:

$$\beta_1^* = -W\left(-\frac{p_j}{p_l(1+m)e}\right), \tag{74}$$

where $\beta_1^* = t_j/\sigma^2(1-t_j)^2$. □

Note that Proposition C.3 applies to both origin- and simplex-centered priors.

Corollary C.1 (Attractors Form in Order)**.** *Let p_l be the largest target probability. For any $j, k \neq l$, if $p_j > p_k$, then $t_j < t_k$. That is, attractor A_j forms earlier than attractor A_k.*

Proof. For the right-hand side of (73), there are two choices of β_1^*, one greater than 1 and the other less than 1. We argue that we select the solution $\beta_1^* > 1$. The Jacobian of the vector field $b_t(x)$ in equation (38) is

$$J(x) = \frac{1}{1-t}\left(J\left(\operatorname{softmax}\left(\log p_i + \frac{tx_i}{\sigma^2(1-t)^2} - \frac{tc_i}{\sigma^2(1-t)}\right)\right) - I\right) \tag{75}$$

where the Jacobian of the softmax vector field can be computed and simplified using (38).

$$J(x) = \frac{1}{1-t}(\beta_1(\operatorname{diag} x - xx^\top) - I) \propto \beta_1(\operatorname{diag} x - xx^\top) - I, \tag{76}$$

where the positive coefficient $1/(1-t)$ is irrelevant. At the time t_j when the j-th attractor appears, the coordinates of the attractor have

$$\det J(x) = 0. \tag{77}$$

Consider the map $F : \mathbb{R}^V \times [0,1] \to \mathbb{R}^V$ via

$$F : (x, \beta_1) \to b_{t(\beta_1)}(x), \tag{78}$$

where $t(\beta_1)$ expresses t in terms of $\beta_1 = \frac{t}{\sigma^2(1-t)^2}$. The Jacobian is the x-derivative of the map $F(x, \beta_1)$. If at β_1^* and the attractor $x(\beta_1^*)$, the determinant

$$\det J(x) = \det D_x F(x, \beta_1) \neq 0, \tag{79}$$

then by Inverse Function Theorem, there exists a neighbourhood B around β_1^* and an neighbourhood U around x^* such that $x(\beta_1) : B \to U$ is a differentiable and satisfies

$$F(x(\beta_1), \beta_1) = 0, \tag{80}$$

for $\beta_1 \in B$. In other words, $x(\beta_1)$ is a family of fixed points of $b_{t(\beta_1)}(x)$. There is a lower time $\beta_1' = \inf B$ that is smaller than the threshold β_1^*, contradicting the assumption that β_1^* corresponds to the smallest time for attractor A_j to exist. Therefore $\det J(x) = 0$ at the threshold β_1^* for attractor A_j.

Write $M = \operatorname{diag} x - xx^\top$, then according to equation (76), the matrix $\beta_1 M - I$ is singular at β_1^* and attractor $x = x(\beta_1^*)$. Therefore, $1/\beta_1^*$ is an eigenvalue of the matrix M. It is well-known that the largest eigenvalue of a matrix of the form M is upper-bounded by $\max_i x_i$, and we have

$$\frac{1}{\beta_1^*} \leq \max_i x_i \leq 1. \tag{81}$$

We have shown that $\beta_1^* \geq 1$. In this range, the mapping $\beta_1 \to \beta_1 - \log \beta_1$ is increasing. In equation (53), a larger p_j corresponds to a smaller β_1^*, and thus a smaller appearance time t_j. □

Proposition C.4 (Most Attractors Form Late). *Let the prior be standard Gaussian $\mathcal{N}(c, \mathbb{I})$ for c subject to Assumption C.1 (i). Except the dominant attractor A_l, no other attractors form earlier than $t = \frac{1}{2}$.*

Proof. We show that $t_j \geq \frac{1}{2}$ for all $j \neq l$ by a closer look at equation (81). The eigenvalue $1/\beta_1^*$ of $M = \operatorname{diag} x - xx^\top$ is smaller than the largest eigenvalue $\lambda_{\max}$ which we can write as

$$\frac{1}{\beta_1^*} \leq \lambda_{\max} = \max_{\|v\|=1} v^\top M v = \max_{\|v\|=1} \operatorname{Var}_x(v). \tag{82}$$

Where $x \in \Delta^{V-1}$ is a distribution and $\operatorname{Var}_x(v)$ takes the variance of v over x. Let $a = \max_i x_i$, $b = \min_i x_i$ and $c = (a+b)/2$ be the midpoint. By the bias-variance decomposition

$$\mathbb{E}_x[(v-c)^2] = \operatorname{Var}_x(v) + (\mathbb{E}_x[v] - c)^2, \tag{83}$$

and the variance is bounded by

$$\operatorname{Var}_x(v) \leq \mathbb{E}_x\left[(v-c)^2\right] = \sum_i x_i (v_i - c)^2 \leq \sum_i x_i \left(\frac{a-b}{2}\right)^2 \leq \frac{a^2+b^2}{2} \leq \frac{\|v\|^2}{2} = \frac{1}{2}. \tag{84}$$

Combining equation (84) and (82) gives $\beta_1^* \geq 2$. When the prior has variance $\sigma^2 = 1$, we have $t_j \geq 1/2$. □

Proposition C.3 and Corollary C.1 show that for target distribution $(p_j)_{j=1,\ldots,V}$, there are V attractors, one corresponding to each vertex. The larger the probability p_j for vertex j, the earlier the attractor A_j forms. From time $t = 0$ to 1, attractors form one by one in the interior of the simplex. By Proposition C.2, the first formed attractor corresponds to the vertex/token of the highest probability p_j. Once an attractor A_j forms, it moves towards vertex j according to Lemma C.2 (i) and (ii). A trajectory that samples a low-probability vertex is pulled away at early times t by the attractors that form earlier, which correspond to high-probability vertices/tokens.

Figure 4 illustrates this on a four-vertex example: the second attractor appears only after $t = 0.5$ despite its probability 0.38 being nearly equal to the dominant 0.4, so the dominant basin is the only one present for the first half of the flow. Figure 5 confirms the picture at scale: the predicted appearance times (53) match simulation up to $V = 100$ (Figure 5A), no non-dominant attractor forms before $t = 0.5$ in any simulation up to $V = 5000$ (Figure 5B), and among non-dominant attractors, larger p_j form earlier (Figure 5C).

C.1.3 Continuity of the Autonomous Velocity

Proposition C.5 (Continuity of the Autonomous Velocity Field). *Let μ_0 be a prior on $\mathbb{R}^V$ that decays exponentially towards infinity. Let the interpolant be $I_t = \alpha_t x_0 + \beta_t x_1$. Then the autonomous velocity $b(x)$ of Proposition 3.1 is continuous at the vertices when $\dot{\alpha}_t = 0$ at $t = 1$. That is $b(x) \to 0$ as $x \to e_i$ for any vertex e_i.*

Proof. We will use Proposition C.6 to show that for $x \to e_i$ for vertex e_i, the autonomous velocity is bounded by

$$\|b(x)\| \leq \frac{\|J^{(i)}(x)\| + \|\sum_{j \neq i} J^{(j)}(x)\|}{|\nu^{(i)}(x)|} \to 0. \tag{85}$$

Let $x \in B_r(e_i)$ the ball of radius r around e_i where we take $r \to 0$. Let $\epsilon_i = x - e_i$ and $\hat{\epsilon}_i$ be the normalized ϵ. The length scale in the integral formula (121) and (123) is

$$v = \frac{r}{\alpha_t} \to \infty, \tag{86}$$

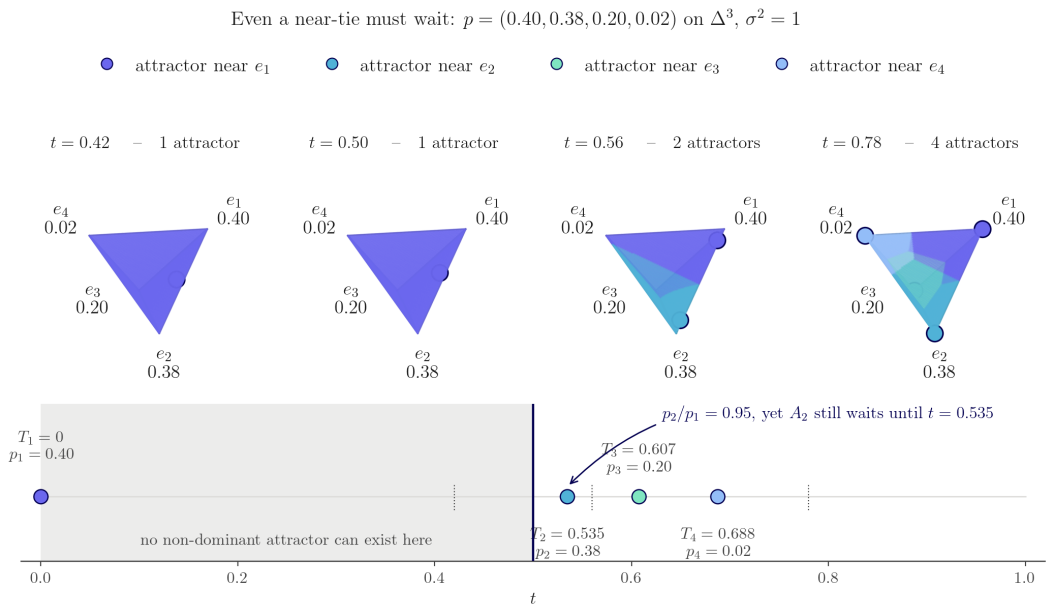

Figure 4: For target probability $p = (0.4, 0.38, 0.2, 0.02)$, the attractor corresponding to the vertex with probability 0.38 forms later than $t = 0.5$, even though the probability 0.38 is only slightly smaller than the largest probability.

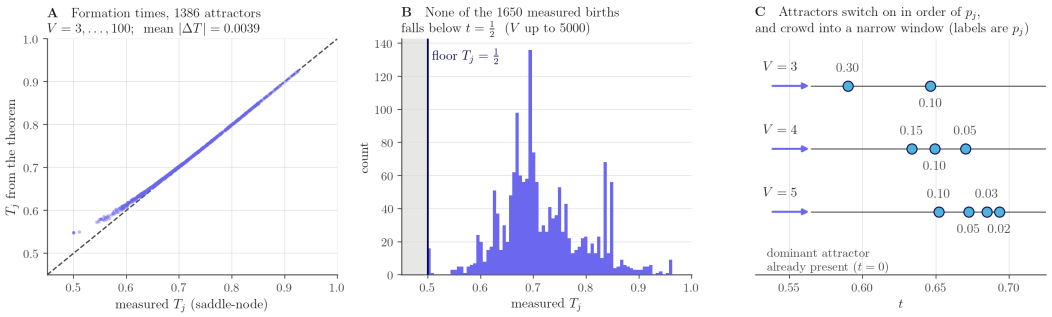

Figure 5: **(A)** Formula (53) predicts the attractor appearance times t_j agrees with simulations. The attractors are simulated up to dimension 100. **(B)** No simulated non-dominant attractors up to dimension 5000 form earlier than $t = 0.5$. **(C)** Non-dominant attractors form earlier for larger probabilities p_j.

as $t \to 1$. The prior μ_0 is evaluated at far-field $e_i + v\,\hat{\epsilon}_i$. Rewrite the integrals $J^{(i)}(x)$ and $\nu^i(x)$ in terms of the new variables v and r we have

$$J^{(i)}(x) = -p_i\, r^{-(V-1)} \hat{\epsilon}_i \int_r^\infty v^{V-1} \mu_0(e_i + v\,\hat{\epsilon}_i)\, dv; \tag{87}$$

$$\nu^{(i)}(x) = p_i\, r^{-(V-1)} \int_r^\infty \frac{v^{V-2} \mu_0(e_i + v\,\hat{\epsilon}_i)}{-\dot{\alpha}_{t(v)}}\, dv, \tag{88}$$

where $t(v)$ expresses time t in terms of v. Since μ_0 decays exponentially towards infinity, there exists an exponential decay that bounds $\mu_0(x) < Ce^{-\lambda x}$ for some $C, \lambda > 0$ for all x. Up to relaxing v_0 to a looser bound v_1, the integral in $J^{(i)}(x)$ is upper bounded by

$$M = \sup_{\|\hat{\epsilon}_i\|=1} \int_0^\infty v^{V-1} \mu_0(e_i + v\,\hat{\epsilon}_i)\, dv < \sup_{\|\hat{\epsilon}_i\|=1} \left\{ \int_0^{v_1} v^{V-1} \mu_0(e_i + v\,\hat{\epsilon}_i)\, dv \right\} + \int_{v_1}^\infty C v^{V-1} e^{-\lambda v}\, dv < \infty. \tag{89}$$

and $J^{(i)}(x)$ is upper bounded by $r^{V-1}\|J^{(i)}(x)\| \leq p_i M$. Now we show that for $j \neq i$, the current $J^{(j)}(x)$ is independent of r, the distance to the vertex e_i which x is approaching. Since x is approaching e_i, the distance from x to vertex e_j is lower bounded by some $l > 0$. Bound $\mu_0(e_j + \frac{\epsilon_i}{\alpha_t}) \leq Ce^{\lambda}e^{-\lambda l/\alpha_t}$ and let L be the upper bound of $|\dot{\alpha}_t| < L$.

$$\|J^{(j)}(x)\| \leq p_j L \int_0^1 \alpha_t^{-(V-1)} Ce^{\lambda}e^{-\lambda l/\alpha_t}\, dt = M_j < \infty, \tag{90}$$

where M_j is independent of x.

Now we lower bound the denominator of equation (85), $\nu^{(i)}(x)$. Define function $h(\delta)$ such that

$$h(\delta) := \sup\{-\dot{\alpha}_t \mid \alpha_t \leq \delta\}. \tag{91}$$

As $\delta \to 0$, we have $h(\delta) \to -\dot{\alpha}_1$, the derivative of α_t evaluated at $t = 1$. The denominator $\nu^{(i)}(x)$ is bounded by

$$r^{V-1}\nu^{(i)}(x) \geq \frac{p_i}{h(r)} \int_1^{\infty} v^{V-2}\mu_0(e_i + v\,\hat{\epsilon}_i)\, dv \geq \frac{p_i\, c_0}{h(r)}, \tag{92}$$

where c_0 is defined as

$$c_0 = \inf_{\|\hat{\epsilon}_i\|=1} \int_1^{\infty} v^{V-2}\mu_0(e_i + v\,\hat{\epsilon}_i)\, dv. \tag{93}$$

The infimum is taken for positive values over a compact set $\{\hat{\epsilon_i} \mid \|\hat{\epsilon_i}\| = 1\}$ and thus exists and is positive. Combining equation (89), (90) and (92), we have the final bound of the autonomous velocity

$$\|b(x)\| \leq \frac{p_i M r^{-(V-1)} + \sum_{j\neq i} M_j}{p_i c_0 r^{-(V-1)}/h(r)} = \frac{h(r)}{p_i c_0}\left(p_i M + \sum_{j\neq i} M_j r^{V-1}\right) \to 0, \tag{94}$$

when $h(r) \to 0$ as $r \to 0$ which happens when $\dot{\alpha}_t \to 0$ as $t \to 1$. □

C.2 Deriving the Phase Transition and Commitment Time

Theorem 4.1 (Phase Transition and Commitment Time)**.** *The phase transition where the interpolant (1) commits to a vertex occurs at time:*

$$t^{\star} = 1 - \left(1 + \sigma\sqrt{2\log V}\right)^{-1/a} \tag{17}$$

where σ is the noise scale of the x_0, and a is the exponent on $\alpha_t = (1-t)^a$ and V is the vocabulary size defining the simplex Δ^{V-1}.

Proof. We want to determine the optimal t_{anchor} for the anchor loss:

$$\mathcal{L}_{\text{anchor}} := \sum_{\ell} \mathbf{1}[t > t_{\text{anchor}}]\text{CE}(T(I_t)_{\ell}, x_{1,\ell}) \tag{95}$$

which preserves the diversity of generations. This means finding the smallest t_{anchor} such that $[t_{\text{anchor}}, 1]$ is after the phase transition has occurred.

Bayes Posterior. Since the transport map is $T_\theta(\cdot) = \text{softmax}(f_\theta(\cdot))$ acting on an intermediate state $x = I_t \in \mathbb{R}^d$, independent of time, the minimizer of $\mathcal{L}_{\text{anchor}}$ given x is the Bayes posterior:

$$T^{\star}(x) = \mathbb{E}[x_1 | I_t = x, t \geq t_{\text{anchor}}] \tag{96}$$

This raises the question: *for which t_{anchor} is the interpolant at I_t committed to the target x_1?* This can also be thought of as identifying the phase transition time in the autonomous flow.

Gibbs measure definition. Given $x_1 = e_k$ and $x_0 = z \sim \mathcal{N}(0, \sigma^2 I_V)$ with noise scale σ, the interpolant is given by:

$$x = I_t = (1-\alpha_t)e_k + \alpha_t z, \quad x|j = k \sim \mathcal{N}((1-\alpha_t)e_j, \alpha_t^2\sigma^2 I_V) \tag{97}$$

By Bayes' rule:

$$p(j=k|x) = \frac{\pi_j p(x|j=k)}{\sum_l \pi_l p(x|j=l)} \tag{98}$$

$$p(x|j=k) = \mathcal{N}((1-\alpha_t)e_j, \alpha_t^2\sigma^2 I_V) = (2\pi\alpha^2\sigma^2)^{-V/2} \exp\left(-\frac{\|x-(1-\alpha)e_j\|^2}{2\alpha^2\sigma^2}\right) \tag{99}$$

Since $(2\pi\alpha^2\sigma^2)^{-V/2}$ doesn't depend on j, it cancels in $p(j=k|x)$. Given $\langle x, e_j\rangle = x_j$ and $\|e_j\|^2$, we can expand the squared distance as:

$$\|x-(1-\alpha)e_j\|^2 = \|x\|^2 - 2(1-\alpha)x_j + (1-\alpha)^2 \tag{100}$$

The j-independent terms cancel, and $p(x|j=k)$ reduces to:

$$p(x|j=k) \propto \exp\left(\frac{(1-\alpha)x_j}{\alpha^2\sigma^2}\right) =: e^{\beta(t)x_j}, \quad \beta(t) := \frac{1-\alpha}{\alpha^2\sigma^2} \tag{101}$$

Substituting into the posterior gives:

$$p(j=k|x) = \frac{\pi_j e^{\beta(t)x_j}}{\sum_l \pi_l e^{\beta(t)x_l}} \tag{102}$$

which is the Gibbs or Boltzmann distribution over vocabulary V at inverse temperature $\beta(t)$, tilted by the prior distribution π, which we assume to be uniform $\pi_j = 1/V$ for simplicity. As $t \to 1$ and $\alpha \to 0$, we have $\beta \to \infty$ and the posterior converges to the Dirac delta on the vertex $p(\cdot|x) \to \delta_k$. As $t \to 0$ and $\alpha \to 1$, we have $\beta \to 0$ and the posterior converges to the prior distribution $p(\cdot|x) \to \pi$.

Signal-to-Noise Ratio. The interpolant is split into a data component and a noise component:

$$I_t = \underbrace{(1-\alpha)e_k}_{\text{signal}} + \underbrace{\alpha z}_{\text{noise}}, \quad \alpha = \alpha_t = (1-t)^a, z \sim \mathcal{N}(0, \sigma^2 I_V) \tag{103}$$

where the signal is on the coordinate k scaled with amplitude $(1-\alpha)$ and the noise spreads over all V coordinates with standard deviation $\alpha\sigma$. So, the signal-to-noise ratio (SNR) per coordinate is:

$$\rho := \rho(t) = \frac{\text{signal amplitude}}{\text{noise std}} = \frac{1-\alpha}{\alpha\sigma} \tag{104}$$

which can be interpreted as how much larger the signal is above the noise floor at time t. For $t \to 1$ near the data, $\rho \to \infty$ and the target token is obvious, and for $t \to 0$ near initial noise, $\rho \to 0$ and the target is unknown. The SNR ρ is monotonically increasing in t, and σ and α determine how fast it increases.

Per-Particle Energies. Since (102) is a Boltzmann distribution, we can define the energy U_j for the vertex j as:

$$U_j := \beta(t)x_j = \beta(t)(1-\alpha)\mathbf{1}[j=k] + \beta(t)\alpha z_j \tag{105}$$

With SNR $\rho = (1-\alpha)/\alpha\sigma$ and $\beta(t) = (1-\alpha)/\alpha^2\sigma^2$, we can expand the energy as:

$$\begin{aligned} U_j &= \frac{(1-\alpha)^2}{\alpha^2\sigma^2}\mathbf{1}[j=k] + \frac{1-\alpha}{\alpha\sigma^2}z_j \\ &= \underbrace{\left(\frac{1-\alpha}{\alpha\sigma}\right)^2}_{=\rho^2}\mathbf{1}[j=k] + \underbrace{\frac{1-\alpha}{\alpha\sigma^2}z_j}_{=:h_j} \end{aligned} \tag{106}$$

Since $z_j \sim \mathcal{N}(0, \sigma^2)$, the variance of h_j is:

$$\text{Var}(h_j) = \left(\frac{1-\alpha}{\alpha\sigma^2}\right)^2 \sigma^2 = \left(\frac{1-\alpha}{\alpha\sigma}\right)^2 = \rho^2 \tag{107}$$

So the energy of the posterior only depends on the SNR ρ:

$$U_j = \rho^2\mathbf{1}[j=k] + h_j, \quad h_j \sim \mathcal{N}(0, \rho^2) \tag{108}$$

Connection to Random Energy Model (REM). This has a nice connection with the Random Energy Model (REM) (Derrida, 1980) in statistical physics, which describes a system with M states with i.i.d. random energies and partition function (normalization constant) $Z = \sum_i e^{-\beta_{\text{phys}}\mathcal{E}_i}$. The normalization of our posterior $p(j = k|x)$ is:

$$Z = \sum_{j=1}^{V} e^{U_j} = \underbrace{e^{\rho^2 + h_k}}_{\text{target signal}} + \underbrace{\sum_{j \neq k} e^{h_j}}_{=:S} \tag{109}$$

where the sum of all non-target vertices S is the REM partition function with $M = V - 1$ states, i.i.d. Gaussian energies $h_j \sim \mathcal{N}(0, \rho^2)$ at physical temperature $\beta_{\text{phys}} = 1$. Then, the probability of recovering $j = k$ from x can be interpreted as whether the energy of state k outweighs the competing energies $j \neq k$.

Phase Transition Point. Now, we want to find the point where the target energy dominates. For $M = V - 1$ states with i.i.d. $\mathcal{N}(0, \rho^2)$ energies, we can count how many exceed a level $\bar{h}$:

$$\mathbb{E}[\#\{j : h_j \geq \bar{h}\}] = M\text{Pr}(h_1 > \bar{h}) \approx M\left(\frac{\rho}{\bar{h}\sqrt{2\pi}}\exp\left(\frac{-\bar{h}^2}{2\rho^2}\right)\right) \approx \exp\left(\log M - \frac{\bar{h}^2}{2\rho^2}\right) \tag{110}$$

The maximum non-target energy occurs when the expected count is equal to 1, i.e., $\log M = \frac{\bar{h}^2}{2\rho^2} \implies h^\star = \rho\sqrt{2\log M}$. Given $M = V - 1$, $\log M \approx \log V$ and:

$$\max_j h_j \approx \rho\sqrt{2\log M} \approx \rho\sqrt{2\log V} \tag{111}$$

Since the target coordinate has energy ρ^2, recovery of the target occurs when ρ^2 surpasses the largest non-target energy $\rho\sqrt{2\log V}$. The critical point is when $\rho^2 = \rho\sqrt{2\log V}$, which for $\rho > 0$ yields:

$$\rho = \sqrt{2\log V} \tag{112}$$

We can define:

$$\theta := \frac{\rho}{\sqrt{2\log V}} \tag{113}$$

so that the critical point occurs when $\theta = 1$.

Argmax Recovery. Taking the argmax coordinate $\hat{k} = \arg\max_j p(k = j|x) = \arg\max_j U_j$ also leads to the critical point. The argmax coordinate is correct (i.e., $\hat{k} = k$) iff the energy of the target state k dominates every non-target state:

$$U_k > \max_{j \neq k} U_j \iff \rho^2 + h_k > \max_{j \neq k} h_j \tag{114}$$

where $h_k \sim \mathcal{N}(0, \rho^2)$ is negligible given ρ^2. Then, we can count the number of non-targets that dominate the target as:

$$\mathbb{E}[\#\{j \neq k : h_j > \rho^2\}] \approx Ve^{-\rho^2/2} = \exp((1 - \theta^2)\log V) = V^{1-\theta^2} \tag{115}$$

which recovers the same critical point:

(i) $\theta > 1$: $V^{1-\theta^2} \to 0$ and none of the non-target states dominate the target, and the argmax is correct $\hat{k} = k$.

(ii) $\theta < 1$: $V^{1-\theta^2} \to \infty$ and many non-targets dominate the target and $\hat{k}$ is a yields a random coordinate.

The critical point occurs when $V^{1-\theta^2} = 1 \iff \theta = 1$ and the target signal equals the maximum non-target $\rho = \sqrt{2\log V}$.

Sharpness of Phase Transition. The maximum of V Gaussians concentrates at $\max_{j \neq k} h_j = \rho\sqrt{2\log V} + \mathcal{O}(\rho/\sqrt{\log V})$ with Gumbel fluctuations. Writing $\rho = \sqrt{2\log V}(1 + \delta)$ and $\theta = 1 + \delta$, the exponent becomes $1 - \theta^2 \approx -2\delta$ and the probability of error is approximately:

$$\text{Pr(error)} \approx V^{-2\delta} = e^{-2\delta\log V} \tag{116}$$

	LM1B	OpenWebText	GSM8K	Sudoku
Vocabulary $\lvert V\rvert$	30,522	50,257	49,152	12
Noise scale σ	1.0	1.0	1.0	1.0
Schedule exponent a	1	1	1	1
$\sqrt{2\log\lvert V\rvert}$	4.545	4.653	4.648	2.229
Optimal t^*_{anchor}	0.820	0.823	0.823	0.690

Table 4: **Optimal anchor threshold t^*_{anchor} per task.** Computed from the phase transition $\rho(t^*) = \sqrt{2\log|V|}$, inverted through the schedule $\alpha_t = (1-t)^a$ to $t^*_{\text{anchor}} = 1 - \left(1 + \sigma\sqrt{2\log|V|}\right)^{-1/a}$. All tasks use $\sigma = 1$ and $a = 1$.

which is near 0 just above zero (i.e., $\delta > 0$) and huge just below $\delta < 0$. The phase transition occurs within the window when $2\delta \log V = \mathcal{O}(1)$ of width $\delta = \mathcal{O}(1/\log V)$ which goes to zero as $V \to \infty$. Therefore, the probability curve sharpens to a step function as $V \to \infty$:

$$\Pr(\hat{k} = k) \to \mathbf{1}[\rho > \sqrt{2\log V}] \quad \text{as} \quad V \to \infty \tag{117}$$

Optimal t_{anchor}. For $\alpha_t = (1-t)^a$, the SNR is:

$$\rho(t) = \frac{1-(1-t)^a}{(1-t)^a\sigma} \tag{118}$$

is monotone in t. Let $t^\star$ denote the optimal t_{anchor}, we can invert $\rho(t^\star) = \sqrt{2\log V}$ to get the optimal $\alpha^\star$:

$$\sqrt{2\log V} = \frac{1-\alpha^\star}{\alpha^\star\sigma} \implies \alpha^\star = \frac{1}{1+\sigma\sqrt{2\log V}} \tag{119}$$

and the optimal $t^\star$:

$$\boxed{t^\star = 1 - \left(1 + \sigma\sqrt{2\log V}\right)^{-1/a}} \tag{120}$$

which concludes the proof. □

C.2.1 Training Implications

Since the simplex dimension appears only in the logarithm $\sqrt{2\log V}$, when V becomes large, large increases in V result in small increases in $t^\star$ (Figure 2D). The exponent on the interpolant schedule a is inversely related to $t^\star$ and moves the curve significantly for $a \in \{1, 2, 3\}$ as shown in Figure 2D and H. Finally, increasing the scale σ of the prior Gaussian $\mu_0 = \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_V)^{\otimes L}$ results in a much larger increase in $t^\star$, since the initial point injects more uncertainty, resulting in a later commitment time. We show the impact of these parameters empirically in App E.

C.3 Occupation Measure of Autonomous Flow on the Simplex

Proposition C.6 (Explicit form of ν and j on simplex)**.** *Let $x_0 \sim \mu_0$, with μ_0 being a prior density on $\mathbb{R}^V$ that decays exponentially towards infinity, and let $x_1 \sim \mu_1 := \sum_{j=1}^V p_j \delta_{e_j}$ be independent of x_0, where $\{e_1, \ldots, e_V\}$ are the vertices of the simplex Δ^{V-1}. Let $I_t = \alpha_t x_0 + \beta_t x_1$ with $\alpha_t + \beta_t = 1$ and $\alpha_0 = 1$, $\alpha_1 = 0$. Fix $x \notin \{e_1, \ldots, e_V\}$ and let $\epsilon_j := x - e_j$. We have the per-vertex occupancy $\nu^{(j)}(x)$ and current $J^{(j)}(x)$ are given by:*

$$\nu^{(j)}(x) = p_j \int_0^1 \alpha_t^{-V}\, \mu_0\left(\frac{\epsilon_j + \alpha_t e_j}{\alpha_t}\right) dt \tag{121}$$

$$J^{(j)}(x) = p_j \int_0^1 \alpha_t^{-V-1} \dot{\alpha}_t\, \epsilon_j\, \mu_0\left(\frac{\epsilon_j + \alpha_t e_j}{\alpha_t}\right) dt, \tag{122}$$

and for $\nu(x) > 0$, the autonomous drift $b = j/\nu$ takes the form:

$$b(x) = \frac{\sum_j J^{(j)}(x)}{\sum_j \nu^{(j)}(x)} \tag{123}$$

Proof. For categorical distributions on the simplex, we can write the target distribution as a weighted mixture of point masses $\mu_1 = \sum_j p_j \delta_{x_j}$ where $x_j = e_j$ and $e_1, \ldots, e_d$ are the d vertices of the simplex Δ^{d-1}, each with probability p_j.

Rather than taking a ratio of measures, we define μ_t and j_t weakly so that the per-vertex decomposition of the autonomous flow can be derived via expectations. For any test function $\varphi \in C^\infty(\mathbb{R}^V)$, we write:

$$\int \varphi(x)\mu_t(x)dx = \mathbb{E}\left[\varphi(I_t)\right], \qquad \int \varphi(x) j_t(x)dx = \mathbb{E}\left[\dot{I}_t \varphi(I_t)\right] \tag{124}$$

which is consistent with the definition $b_t = j_t/\mu_t$ since taking the ratio or the right-hand identities yields $\mathbb{E}[\dot{I}_t | I_t = x]\mu_t(x)$ when tested against φ. Restricting $x_1 = e_j$ gives us the per-vertex definitions of $\mu_t^{(j)}$ and $J_t^{(j)}$ as:

$$\int \varphi(x)\mu_t^{(j)}(x)dx = \mathbb{E}\left[\varphi(I_t)\mathbf{1}_{\{x_1 = e_j\}}\right], \qquad \int \varphi(x) J_t^{(j)}(x)dx = \mathbb{E}\left[\dot{I}_t \varphi(I_t)\mathbf{1}_{\{x_1 = e_j\}}\right] \tag{125}$$

with the occupancy $\nu^{(j)}(x) = \int_0^1 \mu_t^{(j)}(x)dt$ and autonomous current $J^{(j)}(x) = \int_0^1 j_t^{(j)}(x)dt$.

Step 1: Per-vertex occupancy. Since μ_1 is a sum of point masses, ν and j can be decomposed for each vertex $\nu^{(j)}$ and $J^{(j)}$. Therefore, we can consider one atom first. If we fix atom j, then $x_1 = e_j$ and the only randomness in the interpolant is $x_0 \sim \mathcal{N}(0, I_d)$. The interpolant at time t is given by:

$$I_t = \alpha_t x_0 + \beta_t e_j \tag{126}$$

Fixing $I_t = x$, we can solve for $x_0(x,t)$ to get:

$$x_0(x,t) = \frac{x - \beta_t e_j}{\alpha_t} = \frac{x - (1-\alpha_t)e_j}{\alpha_t} = \frac{\epsilon_j + \alpha_t e_j}{\alpha_t}, \tag{127}$$

where we have denoted $\epsilon_j = x - e_j$ and substituted $\beta_t = 1 - \alpha_t$. Substituting into the Gaussian density of x_0 gives a factor of:

$$G_t(x) = \mu_0\left(x_0(x,t)\right) = \mu_0\left(\frac{\epsilon_j + \alpha_t e_j}{\alpha_t}\right) \tag{128}$$

Since the map $x_0(x,t) \mapsto I_t = \alpha_t x_0 + \beta_t e_j$ is just a scaling of x_0 by α_t and a constant shift, its derivative is $\alpha_t I_d$, and the density transforms via the inverse Jacobian α_t^{-d}. Since the probability of selecting vertex j is p_j, the pointwise density $\mu_t^{(j)}(x)$ at x at time t is:

$$\mu_t^{(j)}(x) = p_j \alpha_t^{-V} G_t(x) = p_j\, \alpha_t^{-V} \mu_0\left(\frac{\epsilon_j + \alpha_t\, e_j}{\alpha_t}\right) \tag{129}$$

Then, the occupancy measure $\nu^{(j)}(x)$ is just $\mu_t^{(j)}(x)$ integrated over time $t \in [0,1]$ given by:

$$\nu^{(j)}(x) = \int_0^1 \mu_t^{(j)}(x)dt = p_j \int_0^1 \alpha_t^{-V} \mu_0\left(\frac{\epsilon_j + \alpha_t\, e_j}{\alpha_t}\right) dt \tag{130}$$

Since $\alpha_t \to 0$ as $t \to 1$, we have $\frac{\epsilon_j + \alpha_t e_j}{\alpha_t}$ goes to infinity as $t \to 1$. Hence the prior $\mu_0(\frac{\epsilon_j + \alpha_t e_j}{\alpha_t})$ decays exponentially as $t \to 1$ offsetting the divergent α_t^{-V}, and the integral in equation (130) is finite.

Step 2: Per-vertex current. Given $x_0(x,t) = e_j$, the velocity $\dot{I}_t$ is determined pointwise by x since x_0 is a function of I_t for all $I_t = x$. Differentiating I_t and substituting $x_0 = (\epsilon_j + \alpha_t e_j)/\alpha_t$, we get:

$$\dot{I}_t = \dot{\alpha}_t x_0 + \dot{\beta}_t e_j = \frac{\dot{\alpha}_t}{\alpha_t}(\epsilon_j + \alpha_t e_j) + \dot{\beta}_t e_j = \frac{\dot{\alpha}_t}{\alpha_t}\,\epsilon_j \tag{131}$$

where we have used $\dot{\alpha}_t + \dot{\beta}_t = 0$. Since $j = b\nu = \int_0^1 b_t \mu_t dt$, we:

$$J^{(j)}(x) = p_j \int_0^1 \alpha_t^{-V} \underbrace{\left(\frac{\dot{\alpha}_t}{\alpha_t}\,\epsilon_j\right)}_{\dot{I}_t} \mu_0\left(\frac{\epsilon_j + \alpha_t\, e_j}{\alpha_t}\right) dt = p_j \int_0^1 \alpha_t^{-V-1} \dot{\alpha}_t\, \epsilon_j\, \mu_0\left(\frac{\epsilon_j + \alpha_t e_j}{\alpha_t}\right) dt, \tag{132}$$

by a similar argument to integral (130), the integral (132) is finite.

Step 3: Summing over vertices. Since μ_1 is supported on $\{e_1, \ldots, e_V\}$ and the events $\{x_1 = e_j\}$ partition the sample space such that $\sum_j \mathbf{1}_{\{x_1 = e_j\}} = 1$ almost surely. For any test function φ, linearity of expectation gives:

$$\sum_j \int \varphi(x) J_t^{(j)}(x) dx = \sum_j \mathbb{E}\left[\dot{I}_t \varphi(I_t) \mathbf{1}_{\{x_1 = e_j\}}\right] = \mathbb{E}\left[\dot{I}_t \varphi(I_t) \sum_j \mathbf{1}_{\{x_1 = e_j\}}\right] = \mathbb{E}\left[\dot{I}_t \varphi(I_t)\right] = \int \varphi j_t dx \tag{133}$$

and since this holds for all $\varphi \in C^\infty(\mathbb{R}^V)$, we can conclude $j_t(x) = \sum_j j_t^{(j)}(x)$-a.e.. Analogously, $\mu_t = \sum_j \mu_t^{(j)}$ by the same argument with $\dot{I}_t$ replaced by 1. Integrating over $t \in [0, 1]$ and exchanging the sum and integral operations, we have:

$$\nu(x) = \int \sum_j \mu_t^{(j)}(x) dt = \sum_j \nu^{(j)}(x), \quad J(x) = \int \sum_j J_t^{(j)}(x) dt = \sum_j J^{(j)}(x) \tag{134}$$

Since $b = j/\nu$, we conclude:

$$b(x) = \frac{\sum_j J^{(j)}(x)}{\sum_j \nu^{(j)}(x)} \tag{135}$$

□

C.4 Autonomous Velocity Vanishes at the Vertices

Theorem C.1 (Autonomous Velocity Vanishes at the Vertices)**.** *Assume the interpolant $\alpha_t \sim (1-t)^a$ as $t \to 1$ with $a \geq 1/V$, and assuming the prior $\mu_0(x)$ tends to zero exponentially as $x \to \infty$, then at the vertices of the simplex, for all $i = 1, \ldots, V$, the autonomous velocity field $b(e_i) = 0$ when evaluated at vertices e_i.*

Proof. For given e_i, we evaluate $J^{(j)}(e_i)$ and $\nu^{(j)}(e_i)$ for each j. We compute $\nu^{(j)}(e_i)$ and $J^{(j)}(e_i)$ for the case $j = i$ and $j \neq i$ separately. First when $j = i$, we have $\epsilon_i = e_i - e_i = 0$ and

$$\nu^{(i)}(e_i) = p_i\, \mu_0(e_i) \int_0^1 \alpha_t^{-V}\, dt \tag{136}$$

For common choices of interpolant where $\alpha_t \sim (1-t)^a$ as $t \to 1$ with $a \geq 1/V$, the integral $\int_0^1 \alpha_t^{-V}\, dt$ diverges, and $\nu^{(i)}(e_i)$ is positive infinity.

When $j \neq i$, we have $\epsilon_j = e_i - e_j = e_{ij}$ where e_{ij} is the simplex edge between vertex i and j.

$$\nu^{(j)}(e_i) = p_j \int_0^1 \alpha_t^{-V} \mu_0\left(\frac{e_{ij}}{\alpha_t} + e_j\right) dt \tag{137}$$

As $t \to 1$, we have $\alpha_t \to 0$. Assuming $\mu_0(x)$ tends to 0 exponentially fast as $x \to \infty$, the integral in (137) is finite. The denominator of $b(e_i)$ is $\nu(x) = \sum_j \nu^{(j)}(x)$. The denominator is infinite.

The numerator of $b(e_i)$ is $j(e_i) = \sum J^{(j)}(e_i)$. Consider when the index $j = i$. Since $\epsilon_i = 0$, we have $j^{(i)}(e_i) = 0$. When the index $j \neq i$, for the same reason as $\nu^{(j)}(e_i)$, we have $J^{(j)}(e_i)$ is a finite number.

Overall, for all $i = 1, \ldots, V$, the autonomous velocity field $b(e_i)$ is a ratio between a sum of finite vectors over positive infinity, and hence $b(e_i) = 0$. □

Remark C.2 (Approach speed and explicit hitting-time bound)**.** *Although the autonomous velocity field vanishes at the vertices, $b(e_j) = 0$ for all $e_j \in M_1$, it does not vanish in the limit approaching them: $\lim_{r\downarrow 0} b(e_j + r\omega) = -\kappa(\omega)\omega$. The approach speed is bounded below by $\kappa_- > 0$, and the hitting time in Theorem 3.1(iii) satisfies $\tau(x_0) \le t_{entry} + 2r_1/\kappa_-$ for a fixed radius r_1, where t_{entry} is the first time the flow enters $B(x^\star(x_0), r_1)$. Both facts are established in the proof below.*

C.5 Convergence of Autonomous Flow on the Simplex

Theorem 3.1 (Convergence of autonomous flow on the simplex)**.** *Let μ_0 be absolutely continuous on $(\mathbb{R}^V)^L$ with continuous, strictly positive density, and let*

$$M_1 = \mathrm{supp}(\mu_1) = (\{e_1, \dots, e_V\})^L \subset (\Delta^{V-1})^L \tag{8}$$

be the finite set of V^L vertex configurations. Then for μ_0-a.e. x_0, the autonomous flow $X_s(x_0)$ satisfies:

- *(i)* ***Absorption****: if $X_{s^\star}(x_0) \in M_1$ for some $s^\star < \infty$, then $X_s(x_0) = X_{s^\star}(x_0)$ for all $s > s^\star$.*
- *(ii)* ***Convergence****: there is $x^\star(x_0) \in M_1$ with $\lim_{s\to\infty} X_s(x_0) = x^\star(x_0)$.*
- *(iii)* ***Finite hitting time****: for the linear interpolant $I_t = (1-t)x_0 + tx_1$, $\tau(x_0) := \inf\{t \ge 0 : X_s(x_0) \in M_1\} < +\infty$. Although the autonomous velocity field vanishes at the vertices, $b(e_j) = 0$ for all $e_j \in M_1$, it does not vanish in the limit approaching them: $\lim_{r\downarrow 0} b(e_j + r\omega) = -\kappa(\omega)\omega$. The approach speed is bounded below by $\kappa_- > 0$ and $\tau(x_0) \le t_{entry} + 2r_1/\kappa_-$ for a fixed radius r_1.*

Proof. **Notation.** Let $e_j \in M_1$ be a vertex on the simplex and the state $x = e_j + r\omega$ and define $\omega := (x - e_j)/|x - e_j|$ as the unit vector pointing toward x from e_j and $r := |x - e_j|$ the distance. If the interpolant is $I_t = (1-t)x_0 + tx_1$, then the time-dependent velocity at x at time t is related to the denoiser $m(t)$ by:

$$b_t(x) = \frac{m_t(x) - x}{1 - t}, \quad m_t(x) = \mathbb{E}[x_1 | I_t = x] \tag{138}$$

To get the time-independent velocity, we take the posterior weight over times t with density proportional to $\sigma^{d-2}\mu_0(e_j + \sigma\omega)$, where $\mu_0(e_j + \sigma\omega)$ is the prior evaluated at the implied initial noisy state $x_0 = (x - te_j)/(1-t) = e_j + \sigma\omega$ given x. Taking the average over the time-dependent speed gives the arrival speed $\kappa(\omega)$ toward vertex e_j along direction ω:

$$\kappa(\omega) := \frac{\int_0^\infty \sigma \cdot \sigma^{d-2}\mu_0(e_j + \sigma\omega)d\sigma}{\int_0^\infty \sigma^{d-2}\mu_0(e_j + \sigma\omega)d\sigma}, \quad 0 < \kappa_- := \min_{|\omega|=1} \kappa(\omega) \le \kappa_+ < \infty \tag{139}$$

where $|b(e_j + r\omega)| \to \kappa(\omega)$ is the speed in direction ω as $r \downarrow 0$, κ_- is the minimum speed over all directions, and κ_+ is the maximum speed over directions. We simplify notation by writing:

$$\kappa_j(\omega) := \frac{N_j(\omega)}{D_j(\omega)}, \quad N_j(\omega) := \int_0^\infty \sigma^{d-1}\mu_0(e_j + \sigma\omega)d\sigma, \quad D_j(\omega) := \int_0^\infty \sigma^{d-2}\mu_0(e_j + \sigma\omega)d\sigma \tag{140}$$

Given $x_1 = e_j$ and $\alpha_t := 1 - t$, we write the marginal distribution at time t as:

$$\mu_t(x) = \sum_{e_l \in M_1} \pi_{e_l} \alpha_t^{-d} \mu_0\left(\frac{x - te_l}{\alpha_t}\right) \tag{141}$$

Autonomous velocity approaching vertex. We first prove a Lemma that gives the speed at which the autonomous flow approaches a vertex.

Lemma C.3 (Autonomous velocity approaching vertex)**.** *Let $e_j \in M_1$ be a vertex on the simplex and $\omega := x - e_j$ with $|\omega| = 1$ be the unit vector pointing in the direction from x to e_j. Then,*

$$\nu(x) = \pi_{e_j} r^{1-d} D_j(\omega)(1 + o(1)) \tag{142}$$

and $b(e_j + \sigma\omega) \to -\kappa_j(\omega)\omega$ in the limit $r \downarrow 0$, uniformly in ω. Therefore, b is discontinuous at e_j, where its value at the vertex is $b(e_j) = 0$ while its limits approaching e_j radially are non-zero and direction-dependent.

Proof. Let $x := e_j + r\omega$ and $\sigma := r/\alpha$. Then, $\alpha = r/\sigma$ and $|d\alpha| = r\sigma^{-2}d\sigma$. The argument in μ_0 in $\mu_t(x)$ (141) for e_j takes the form:

$$\frac{x - te_j}{\alpha_t} = \frac{r\omega + \alpha_t e_j}{\alpha_t} = e_j + \sigma\omega \tag{143}$$

so we can write $\mu_t(x)$ as:

$$\mu_t(x) = \pi_{e_j} \left(\frac{\sigma}{r}\right)^d \mu_0(e_j + \sigma\omega)(1 + O(\alpha)) \tag{144}$$

where $O(\alpha)$ contains the summands for $e_l \neq e_j$ which becomes negligible after integration over $t \in [0, \infty)$. Therefore, the occupation measure $\nu(x)$ is given by:

$$\begin{aligned}
\nu(x) &= \pi_{e_j} \int_0^\infty \left(\frac{\sigma}{r}\right)^d \mu_0(e_j + \sigma\omega) \frac{r d\sigma}{\sigma^2}(1 + o(1)) \\
&= \pi_{e_j} r^{1-d} \underbrace{\int_0^\infty \sigma^{d-2} \mu_0(e_j + \sigma\omega) d\sigma}_{D_j(\omega)} (1 + o(1)) \\
&= \pi_{e_j} r^{1-d} D_j(\omega)(1 + o(1))
\end{aligned} \tag{145}$$

which is the expression in (142). To get $b(x)$, we can express $b_t(x)$ in (138) with $m_t(x) := e_j + O(e^{-c/\alpha^2})$ as:

$$b_t(x) = \frac{e_j - x}{\alpha} + O\left(\alpha^{-1} e^{-c/\alpha^2}\right) = -\frac{r}{\alpha}\omega + o(1) = -\sigma\omega + o(1) \tag{146}$$

where the rescaled radius σ is the speed. Therefore, the numerator of the autonomous flow $b(x)$ is:

$$\int_0^1 \mu_t(x) b_t(x) dt = -\omega \pi_{e_j} r^{1-d} \int_0^\infty \sigma^{d-1} \mu_0(e_j + \sigma\omega) d\sigma (1 + o(1)) = -\omega \pi_{e_j} r^{1-d} N_j(\omega)(1 + o(1)) \tag{147}$$

and dividing by $\nu(x)$ yields $b(x) = -\omega N_j(\omega)/D_j(\omega) + o(1) = -\kappa(\omega)\omega$, which is not dependent on r. □

Now, we prove Proposition 3.1.

By Lemma C.3, set a small constant $r_1 > 0$ such that:

$$\langle b(e_j + r\omega), \omega \rangle \leq -\frac{\kappa_-}{2} \quad \text{for all } e_j \in M_1, |\omega| = 1, 0 < r \leq r_1, \tag{148}$$

so the ball with radius r_1 around each vertex $\{B(e_j, r_1)\}_{e_j \in M_1}$ are pairwise disjoint.

Local absorption in finite time. Suppose that $X_s(x_0) \in B(e_j, r_1)$ for some finite time $s < \infty$ and some $e_j \in M_1$. Let $h(s) := |X_s(x_0) - e_j|$ be the distance from the vertex. Over any time interval where $0 < h \leq r_1$, $h(t)$ is locally Lipschitz and $\dot{h}(s) = \langle b(X_s), (X_s - e_j)/h(s) \rangle \leq -\kappa_-/2$ by (148). Therefore, $h(s)$ decreases at rate at least $\kappa_-/2$ and must vanish at some finite time:

$$\tau \leq s + \frac{2h(s)}{\kappa_-} \leq s + \frac{2r_1}{\kappa_-} < \infty \tag{149}$$

Absorption. From $\dot{h}(s) \leq -\kappa_-/2$, we have shown that $X_{s^\star}(x_0)$ arrives at e_j in some finite time. Now, we argue that for all $s > s^\star$, $X_s(x_0) = e_j$ by contradiction. Suppose there exists $X_s(x_0) \neq e_j$ for $s > s^\star$, where $h(t) \leq r_1$ on $[s^\star, s]$. Then, we have shown $\dot{h}(s) \leq -\kappa_-/2$ a.e. on $\{h > 0\} \cap [s^\star, s]$ and since $h(s)$ is absolutely continuous with $h(s^\star) = 0$, we can write:

$$h(s) = \int_{s^\star}^{s} \dot{h}(u)du \leq -\frac{\kappa_-}{2}(t - t^\star) < 0 \tag{150}$$

which is a negative distance $h(s)$, a contradiction. Thus, there exists no $X_s(x_0)$ that leaves e_j and $X_s(x_0) = e_j$ for all $s > s^\star$.

Lemma C.4 (Absorbance from far field)**.** *For b satisfying the divergence condition in Proposition B.1, $b(x) = -x(1 + o(1))$ for $|x| \to \infty$ so there exists radius $R_0 < \infty$ with $\langle b(x), x\rangle < 0$ for $|x| \geq R_0$. Let $B(0, R_0)$ be the ball with radius R_0 centered at the origin, then $X_s \in B(0, R_0)$ for all $s \geq t_0$ if $X_{t_0} \in B(0, R_0)$ (no points escape the ball) and for every x_0 there is a finite t_0 with $X_s(x_0) \in B(0, R_0)$ for all $s \geq t_0$ (all points end up in the ball).*

Proof. Since $b(x) = -x(1 + o(1))$ as $|x| \to \infty$, by fixing R_0, we have:

$$\langle b(x), x\rangle \leq -\tfrac{1}{2}|x|^2 \quad \text{for } |x| \geq R_0 \tag{151}$$

Let $h(s) := |X_s|$ be the distance from the origin and differentiate:

$$\frac{d}{dt}\tfrac{1}{2}|X_s|^2 = \langle \dot{X}_s, X_s\rangle = \langle b(X_s), X_s\rangle < 0 \quad \text{whenever} \quad |X_s| \geq R_0 \tag{152}$$

where $\langle b(x), x\rangle$ is the radial component of the velocity scaled by $|x|$, which is negative exactly when the field $b(x)$ points inward toward the origin.

No escape. We prove by contradiction. Suppose a point $X_{t_0} \in B(0, R_0)$ escapes to $h(t_1) > R_0$ at some later time $t_1 > t_0$. Let s_1 be the last time between $[t_0, t_1]$ where the trajectory was at radius R_0 or below: $s_1 := \sup\{s \in [t_0, t_1] : h(s) \leq R_0\}$. Then, $h(s_1) = R_0$ by continuity of the trajectory and the distance $h > R_0$ for all $(s_1, t_1]$ by definition of s_1 as the last time where $h \leq R_0$. But by (152), the radius must be decreasing $\frac{d}{ds}\frac{1}{2}h^2 < 0$ whenever $h \geq R_0$ which means $h(t_1) < h(s_1) = R_0$, a contradiction. Therefore, $X_s \in B(0, R_0)$ for all $s \geq t_0$.

Entry in finite time. While $h(s) \geq R_0$, (151) gives $\dot{h}(s) = \langle b(x), x\rangle \leq -\frac{1}{2}h(s)$, so $h(s) \leq |x_0|e^{-s/2}$. Therefore, $h(s) < R_0$ for all $s > t_0 := \max(0, 2\log(|x_0|/R_0)) < \infty$, and by the no escape proof above, the trajectory remains in $B(0, R_0)$ for all times after entry. □

Now, we continue to prove (i) and (ii) of Proposition 3.1.

Convergence. Given that all points in $\{B(e_j, r_1)\}$ are absorbed on the vertices in finite time, we now show that any initial point x_0 converges to M_1. Let $U := \bigcup_{e_j \in M_1} B(e_j, r_1)$ and $E := B(0, R_0) \setminus U$, and define the set of initial points x_0 whose autonomous flow $X_s(x_0)$ does not land in U for all $t \geq 0$:

$$N := \{x_0 : X_s(x_0) \notin U \text{ for all } s \geq 0\} \tag{153}$$

By Lemma C.4, all points x_0 land in the ball $B(0, R_0)$ at finite time t_0 and remain inside the ball for all $s \geq t_0$, so for $x_0 \in N$, the flow trajectory lies in the set E for all $s \geq t_0$. By Lemma C.5 below, we show that the set N is Lebesgue-null $\mathrm{Leb}(N) = 0$ (zero d-dimensional volume) such that $\mu_0 \ll \mathrm{Leb}$ and $\mu_0(N) = 0$.

In the case where $x_0 \notin N$, the autonomous flow enters $B(e_j, r_1)$ for some $e_j \in M_1$ at a finite time s and by (149), it reaches e_j at some time $\tau(x_0) \leq s + 2r_1/\kappa_-$ and by the absorption proof, it remains there for all $t \geq \tau$. Setting $x^\star(x_0) := e_j$ gives $X_s(x_0) = x^\star(x_0)$ for all time $t \geq \tau(x_0)$.

Lemma C.5. $\mathrm{Leb}(N) = 0$ *where* $N := \{x_0 : X_t(x_0) \notin U \text{ for all } t \geq 0\}$

Proof. Given $\nabla \cdot (\nu b) = \mu_0 - \mu_1$ and $\mu_1(E) = 0$, we have:

$$\nabla \cdot b(x) = \frac{\mu_0(x) - \langle \nabla \nu(x), b(x) \rangle}{\nu(x)} \tag{154}$$

for $x \in E$. Let $J_t := \det(DX_t(x))$ be the Jacobian determinant of the autonomous flow $x \mapsto X_t(x)$, which acts like a change-of-variables factor $\mathrm{Leb}(X_t(A)) = \int_A J_t(x)dx$. Along any trajectory of the autonomous flow, we have:

$$\frac{d}{dt}[\nu(X_t(x))J_t] = \mu_0(X_t(x))J_t \tag{155}$$

So for the set $N_n := \{x_0 : X_t(x_0) \notin U \text{ for all } t \geq t_n\}$, the occupation measure of the autonomous flow is $m(t) := \nu(X_t(N_n)) = \int_{N_n} \nu(X_t(x))J_t dx$ for $t \geq t_n$. Differentiating, we have:

$$\dot{m}(t) = \int_{N_n} \mu_0(X_t(x))J_t dx = \int_{X_t(N_n)} \mu_0(y)dy \geq c_0 \mathrm{Leb}(X_t(N_n)) \geq \frac{c_0}{\nu_+} m(t) \tag{156}$$

where we apply the change of variables $y := X_t(x)$, then $\mu_0(E) \geq c_0 > 0$ on E, then $\nu \leq \nu_+$ gives $\nu(A) \leq \nu_+ \mathrm{Leb}(A)$. By Grönwall's inequality, $m(t) \geq m(t_n)\exp(\frac{c_0}{\nu_+}(t - t_n))$ for $t \geq t_n$. Since $X_t(N_n) \subseteq E$ for all $t \geq t_n$ by definition, we have $m(t) \leq \nu(E) \leq \nu_+ \mathrm{Leb}(E) < \infty$. The exponentially growing quantity cannot be bounded by infinity, forcing $m(t_n) = 0$. Since $\nu(E) \geq \nu_- > 0$, we have $\mathrm{Leb}(X_{t_n}(N_n)) = 0$ and X_{t_n} is a diffeomorphism on the neighborhood of E where $b \in C^1$, we conclude $\mathrm{Leb}(N_n) = 0$. □

Finite hitting time. Let $x_0 \notin M_1$ be a point in μ_0 and let t_{entry} be the first time when $X_s(x_0) \in B(x^\star(x_0), r_1)$, which we proved to be finite in the proof of (i). Then, from (149), we have:

$$\tau(x_0) \leq t_{\text{entry}} + \frac{2r_1}{\kappa_-} < \infty \tag{157}$$

and $X_s(x_0) = x^\star(x_0)$ for all $s \geq \tau(x_0)$ by (ii). Note that by Lemma C.3, the trajectory approaching $x^\star(x_0) = e_j$ arrives at speed $-\kappa_j(\omega)\omega$ and stops at e_j where $b(e_j) = 0$ by Theorem C.1. □

C.6 Eulerian and Conservation Equations

Proposition 3.2 (Autonomous transport map solves conservation equation (Lee et al., 2026b)). *Given the autonomous flow velocity field b, the autonomous flow $X_s(x_0)$ defined in (5) solves the Eulerian equation:*

$$\begin{cases} b(x) \cdot \nabla X_s(x) = \partial_s X_s(x), & X_0(x) = x \quad x \notin M_1 \\ X_s(x) = x & x \in M_1 \end{cases} \tag{10}$$

Since the autonomous transport map $T(x)$ defined in (9) is the limit of $X_s(x)$ as $s \to \tau(x)$ with vanishing velocity at the vertices ($\dot{X}_s(e_j) = 0$ for all $j \in \{1, \ldots, V\}$) by Theorem 3.1, $T(x)$ is the unique solution of the conservation equation:

$$\begin{cases} b(x) \cdot \nabla T(x) = 0 & x \notin M_1 \\ T(x) = x & x \in M_1 \end{cases} \tag{11}$$

Proof. We define $M_1 = \{e_1, \ldots, e_V\}$ for the set of vertices of Δ^{V-1} on which the autonomous flow b vanishes by Theorem C.1. This proof follows closely from that of Theorem 2 and Proposition 2 of Lee et al. (2026b), which we restate here in the simplex setting.

Eulerian equation off M_1. Since b has no time dependence, for $x \notin M_1$ and $s, u \geq 0$ with $s + u < \tau(x)$, the uniqueness of the autonomous flow $X_s(x)$ gives the semigroup identity:

$$X_{s+u}(x) = X_u(X_s(x)) \tag{158}$$

By differentiating (158) in s at $s = 0$ and applying the chain rule on the RHS, we get:

$$\partial_s X_s(x) = \nabla X_s(x) \cdot \partial_s X_s(x)|_{s=0} = b(x) \cdot \nabla X_s(x) \tag{159}$$

with $X_0(x) = x$ by definition of the flow. For $x \in M_1$, $b(x) = 0$ by Theorem C.1, so the constant curve $X_s(x) = x$ is the unique solution, which yields (10).

Conservation equation of the transport map. For $x \notin M_1$ and $s \in [0, \tau(x))$, by Proposition 3.1, the autonomous ODE trajectory passing x converges to a vertex, which is also the same trajectory passing $X_s(x)$. Therefore, the transport map T, defined as the long-time limit of the autonomous flow, satisfies:

$$T(X_s(x)) = T(x) \quad \forall s \in [0, \tau(x)) \tag{160}$$

Differentiating (160) in s at $s = 0$ yields $b(x) \cdot \nabla T(x) = 0$, which is the conservation equation in (11). Since $b(x) = 0$ for $x \in M_1$, $T(x) = \lim_{s \to \tau(x)} X_s(x) = x$ satisfies the boundary condition.

Uniqueness of T. Let $\tilde{T}$ be any solution of the conservation equation that is continuous along the flow trajectories. Then $\frac{d}{ds}\tilde{T}(X_s(x)) = b(X_s(x)) \cdot \nabla \tilde{T}(X_s(x)) = 0$ so $s \mapsto \tilde{T}(X_s(x))$ is constant. Letting $s \to \tau(x)$ and using $\tilde{T} = \mathrm{id}$ on M_1 gives $\tilde{T}(x) = \lim_{s \to \tau(x)} X_s(x) = T(x)$. □

D Refinement-in-Loop Details and Additional Results

A core component of our training approach for DBTM is concentrating training time on states that are likely to be seen during inference, and therefore, minimizing the training-inference mismatch. To do this, we mimic the inference procedure of BTM, consisting of clean sequence proposals, renoising of low-quality tokens, and refining the proposal during training using the current model, and minimize the loss on these self-generated rollout states.

D.1 On-policy Rollouts

The standard method of constructing the context-dependent interpolant is to sample the fraction of clean tokens $f \sim \mathcal{U}[0,1]$ and apply an i.i.d. Bernoulli(f) per position to determine whether it is in $\mathcal{C}$. However, this creates a mismatch between training and inference since the context fed back into the model during refinement is not random. Instead, the context is largely dependent on how 'easy' it is to be generated at the initial application of the transport map T_θ and the subsequent refinement maps. To align the context seen during inference to the ones used for training, we apply an **on-policy rollout step** to generate a set of partial context sequences for training by running the same map, renoise, and refine steps as done during inference with a predetermined commitment condition.

Confidence-Based Selection To determine which tokens to commit during the training-time rollouts, we use a confidence-based selection scheme. Given a coupling (x_0, x_1), for a rollout of depth k, let $\hat{x}_{r,k} := \text{renoise}(T_\theta(\hat{x}_{r-1,k}))$ denote the state after r refinement rounds without gradient tracking where $\hat{x}_{0,k} = x_0$ is the initial noise state and $\hat{x}_{k,k}$ is the final clean sequence after k rounds. After each round r, we perform the following steps on the output of the previous round $\hat{x}_{r-1,k}$:

(i) Apply the parameterized transport map to get a clean proposal sequence $T_\theta(\hat{x}_{r-1}, k)$.

(ii) Rank each position ℓ by the model's predicted confidence on the clean token value. Given the one-hot vector x_1^ℓ, compute the confidence at each position ℓ as $q^\ell := \langle T_\theta(\hat{x}_{r-1}, k)^\ell, x_1^\ell \rangle$ to get an ordered set $\mathcal{R}_r$ of the uncommitted positions.

(iii) Since there are k total rounds, each round selects at minimum $n_r = \lceil |\mathcal{R}_r|/(k-r+1) \rceil$ highest confidence tokens to commit.

Since the map T_θ is trained with the partial context interpolant, with more context yielding a sharper prediction task, this selection scheme uses the model's own predictions to distinguish key structural tokens that are easy to predict without context from difficult, highly context-dependent tokens, and trains on the partial context sets it is likely to encounter during inference.

Committed vs. Non-Committed Clean Context We perform two methods of refinement-in-loop training: either the clean context at each rollout step is *committed* or *not committed*. For the committed setting, at each iteration i of the rollout with depth r, we apply the current trained map $T_\theta(\hat{x}_{i-1,r})$, commit all tokens with $q^\ell \geq \kappa$ or at least $n_r = \lceil |\mathcal{R}_r|/(k-r+1) \rceil$ (defined as the set $\Delta\mathcal{C}_r$) where $\mathcal{R}_r$ are the uncommitted tokens, and reset the remaining tokens back to their coupled noise state x_0^ℓ for $\ell \in [L] \setminus \mathcal{C}_r$. After the rollout step, the loss in (24) is computed on only the uncommitted tokens $[L] \setminus \mathcal{C}_r$ at each rollout depth r:

$$\mathcal{L}_{\text{ril-c}}(\theta) := \mathbb{E}_{x_0,x_1}\left[\sum_{k\in\mathcal{K}}\sum_{r=1}^{k}\sum_{\ell\in[L]\setminus\mathcal{C}_r} \text{CE}\left(T_\theta(\hat{x}_{r-1,k})^\ell, x_1^\ell\right)\right] \tag{161}$$

where `ril-c` denotes the loss for the committing rollout setting. We can also define the transport loss (14) and anchor loss (18) on these refinement-generated context sets, with $\ell \in [L] \setminus \mathcal{C}_r$ being the uncommitted tokens that the loss is computed over.

For the non-committed setting, at each iteration i of the rollout with depth r, we apply the current trained map $T_\theta(\hat{x}_{i-1,r})$, but instead *hold the highest confidence tokens in set* $\mathcal{C}_r$, such that they remain in their mapped categorical state $\hat{x}_{i,r}^\ell \leftarrow T_\theta(\hat{x}_{i-1,r})^\ell \in \Delta^{V-1}$ for $\ell \in \mathcal{C}_r$ and can be changed in the next round of refinement. The remaining tokens are similarly reset to their coupled noise state $\hat{x}_{i,r}^\ell \leftarrow x_0^\ell$ for $\ell \in [L] \setminus \mathcal{C}_r$. After rollout, the loss is now computed on *all the tokens in the map output* at each rollout depth r:

$$\mathcal{L}_{\text{ril-nc}}(\theta) := \mathbb{E}_{x_0,x_1}\left[\sum_{k\in\mathcal{K}}\sum_{r=1}^{k}\sum_{\ell\in[L]} \text{CE}\left(T_\theta(\hat{x}_{r-1,k})^\ell, x_1^\ell\right)\right] \tag{162}$$

where the sum is taken over all token positions $\ell \in [L]$ instead of only the uncommitted set and `ril-nc` denotes the loss for the non-committed rollout setting.

Quality Head on Refinement States The states $\hat{x}_{r,k}$ generated by the commit rule (23) are also used to train the quality head:

$$\mathcal{L}_{\text{ril-}\phi}(\phi) := \mathbb{E}_{x_0,x_1}\left[\sum_{k\in\mathcal{K}}\sum_{r=1}^{k}\sum_{\ell\in[L]\setminus\mathcal{C}_r} \text{BCE}\left(\mathbf{1}[\hat{x}_1^\ell = x_1^\ell],\ q_\phi^\ell(T_\theta(\hat{x}_{r-1,k}))\right)\right], \quad \hat{x}_1 = \arg\max(T_\theta(\hat{x}_{r-1,k})) \tag{163}$$

with the argument of q_ϕ detached.

Self-Distillation with Semigroup Loss Since the noise-data coupling is generally not well-defined for unconditional text generation where there is no prompt and the space of possible clean outputs is large, supervising intermediate rollouts only with clean data samples can prevent the model from generating diverse text since it forces intermediate states to produce a single output even when that output is not aligned with its generative trajectory.

Therefore, to encourage coherent yet diverse supervision, we incorporate a form of self-distillation into the objective. After a warm-up stage where the model starts to generate coherent text, the output after k refinement rounds approximates a clean sample from μ_1, and its coupling to the initial noise x_0 is automatic, removing the need for a prescribed noise-data coupling. Writing $\hat{x}_{r,k}$ for the state after r rounds under budget k with $\hat{x}_{0,k} := T_\theta(x_0)$, the *semigroup objective* matches the map at every intermediate round to the final output:

$$\mathcal{L}_{\text{semi-r}}(\theta) := \mathbb{E}_{x_0,x_1}\left[\sum_{k\in\mathcal{K}}\sum_{r=1}^{k}\sum_{\ell} \text{CE}\left(T_\theta(\hat{x}_{r-1,k}), \text{sg}(T_\theta(\hat{x}_{k-1,k}))\right)\right] \tag{164}$$

where $\hat{x}_{k,k} = T_\theta(\hat{x}_{k-1,k})$ is the final clean sequence after k rounds of refinement. This is analogous to the semigroup loss (15) but with the map composed with the renoise operation. Variants restricted to committed or non-committed positions are given by:

$$\mathcal{L}_{\text{soft-ril-c}}(\theta) := \mathbb{E}_{x_0,x_1}\left[\sum_{k\in\mathcal{K}}\sum_{r=1}^{k}\sum_{\ell\in[L]\setminus\mathcal{C}_r} \text{CE}\left(T_\theta(\hat{x}_{r-1,k})^\ell, \text{sg}(T_\theta(\hat{x}_{k-1,k})^\ell)\right)\right] \tag{165}$$

$$\mathcal{L}_{\text{soft-ril-nc}}(\theta) := \mathbb{E}_{x_0,x_1}\left[\sum_{k\in\mathcal{K}}\sum_{r=1}^{k}\sum_{\ell\in[L]} \text{CE}\left(T_\theta(\hat{x}_{r-1,k})^\ell, \text{sg}(T_\theta(\hat{x}_{k-1,k})^\ell)\right)\right] \tag{166}$$

where `soft` denotes that the target can be any categorical distribution on the interior of the simplex rather than only one-hot vectors on the vertices.

D.2 Refinement-in-Loop for Reasoning

Setup In the Sudoku setting, the prior state x_0 is not just Gaussian noise but a 9×9 grid with a set of filled-in numbers as clues depending on the difficulty (easy: 40, medium: 35, hard: 30), which each correspond exactly to one solution x_1. Therefore, given x_0, there is no uncertainty about what the answer x_1 should be, so x_1 should be the only supervision target, and self-distillation is unnecessary, since the self-generated supervision target can only be either equivalent to x_1 or incorrect.

Instead, we test the effect of refinement-in-loop only with the defined (x_0, x_1) data couplings and no self-distillation. We set the maximum depth to $k = 4$ NFEs (initial map + three refinement rounds), and for each round $r \in \{1, \ldots, k\}$ perform refinement by committing the positions with the highest ground-truth confidence, reset the uncommitted positions to their prior state x_0^ℓ for $\ell \in [L] \setminus \mathcal{C}_r$ to form a set of self-rollout states $\hat{x}_{r,k}$ for supervising against the ground truth via the `ril-c` loss in (161). A summary of experiment settings is provided in Table 5.

Results In Table 2, we show that DBTM with refinement-in-loop (DBTM + ril) achieves superior performance against all baselines and difficulty levels with only 4 NFEs, $32\times$ fewer NFEs than autoregressive, discrete diffusion, and continuous diffusion baselines, and matched NFEs with the closest flow map language model baseline, FMLM+ (Agarwal et al., 2026). Notably, on Sudoku Hard, DBTM + ril surpasses FMLM+ by 16.1% at 16 NFEs and 13.4% at 4 NFEs. As shown in Figure 6, DBTM trained on the clean-context interpolant can converge to high solve accuracy without ril training, but DBTM + ril *significantly* accelerates convergence while achieving higher accuracy.

Variable	LM1B	Sudoku
Refinement Depth k	4	4
Supervise Depths (`all` depths to k or `one` randomly sampled)	`all`	`all`
Samples Per Data	1	1
Selection Scheme	confidence on x_1	confidence on x_1
Confidence Threshold κ	0.9	0.9
Noise-Data Coupling	OT coupling	Clues and clean grid + OT coupling
Fresh noise	`off`	`off`
Self-distill	`on`	not necessary given defined clue-solution coupling
Commitment Scheme	`commit/non-commit`	N/A
Soft target	`true`	N/A
Temperature	0.7	N/A
Self-distill start step	100,000	N/A
Teacher refresh frequency (steps)	5000	N/A
Supervise Depths r	$\{0, 1, 2, 3\}$	N/A

Table 5: **Refinement-in-loop training setup for LM1B and Sudoku.** Both tasks roll out to depth k=4 and supervise every depth, committing by confidence at κ=0.9 without fresh noise. LM1B adds self-distillation after a 100k-step warmup with soft targets at temperature 0.7 and a teacher refreshed every 5000 steps; Sudoku needs none, since its clue-solution coupling already fixes the target.

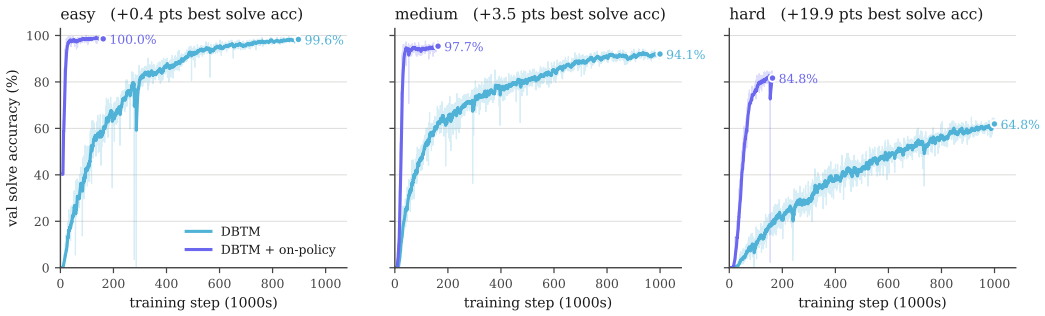

Figure 6: **Sudoku validation solve accuracy with and without refinement-in-loop training.** Exact-solve accuracy on the held-out puzzles over training steps at 4 NFEs. The clean-context training reaches high accuracy on its own, but ril training converges significantly faster and to a higher ceiling on every difficulty.

D.3 Refinement-in-Loop for Unconditional Language Modeling

Setup In the language modeling task, there is no definitive coupling between noise and data, since, given a noise sample x_0, any coherent text sequence is a valid target sample x_1 and there is no *'right'* or *'wrong'* answer like in the reasoning case. Defining a random coupling between noise and data and supervising the on-policy rollouts from the noise to match the data is susceptible to mode collapse, since the noise can be close to generating a completely different, but still coherent, sequence before it is forced to match an arbitrary target. This differs from the base training procedure where the intermediate states are constructed from the interpolant defined as a function of *both* x_0 and x_1.

We avoid mode collapse in two complementary ways. First, we use minibatch optimal transport (Tong et al., 2024) to define the coupling (x_0, x_1) for supervised ril training and test both the *committed* and *non-committed* clean context methods described in App D.1 with either the target x_1 being committed or held at the categorical state at the positions where the map is most confident. Second, we perform self-distillation after a warmup phase in parallel to the data-supervised training as described in App D.1, where the target is a *function* of the noise input x_0, making the self-generated and refined sequence a well-defined target. We use the LM1B dataset as a testbed for these methods and provide a summary of experiment settings in Table 5.

Results In Figure 3 and Table 13, we show that DBTM with refinement-in-loop (commit) achieves the widest Pareto frontier of all methods, with low Gen. PPL across all entropy values from $3.90-4.30$. Notably, DBTM + ril achieves the lowest PPL along the frontier curve across all NFE counts and methods. When comparing commit vs. non-commit ril training, we found that the commit mode consistently achieves lower Gen. PPL across NFE counts and selection thresholds with stable entropy around 4.00 (Table 6). Figure 7 plots the Gen. PPL and sample entropy from small validation batches

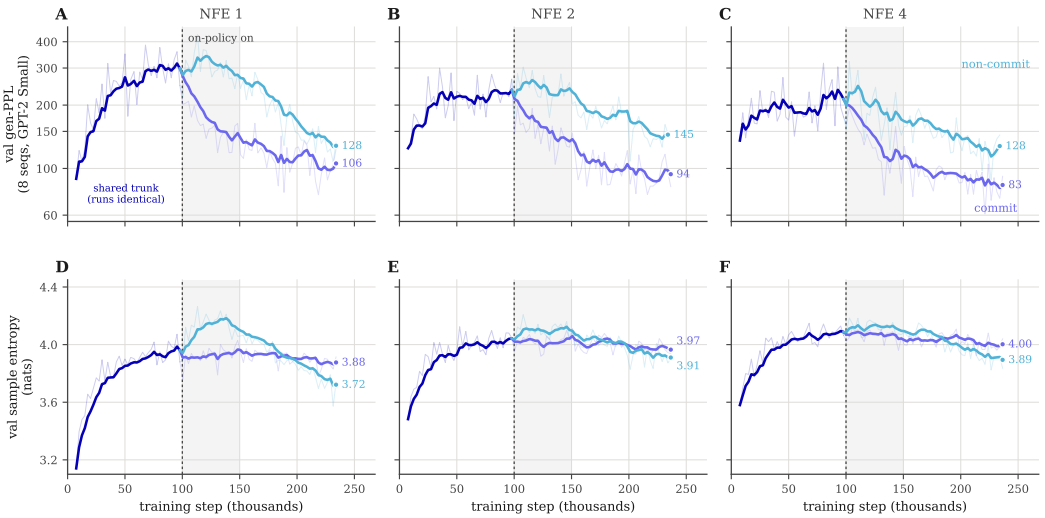

Figure 7: **Effect of refinement-in-loop training on LM1B.** Gen-PPL (GPT-2-small for efficiency) and sample entropy on a single validation batch of 4 sequences over training (not directly comparable to the table values). Gen-PPL rises early as entropy approaches the data entropy, and when on-policy training turns on after the 100K-step warmup, Gen-PPL drops while entropy holds. Panels A–C compare committed (purple) and non-committed (turquoise) rollouts; the committed variant descends faster at every NFE.

over training steps. Visually, Gen. PPL rises early in training as entropy approaches the data entropy, and when on-policy training turns on after the 100K- step warmup, Gen. PPL starts to drop while entropy remains stable. In Figure 7A-C, the commit method (purple curve) shows a steeper drop in Gen. PPL than the non-commit method (turquoise curve).

	commit Commit Threshold κ					**non-commit** Commit Threshold κ				
NFE	0.5	0.7	0.9	0.99	0.999	0.5	0.7	0.9	0.99	0.999
1	86.0 (3.92)	86.0 (3.92)	86.0 (3.92)	86.0 (3.92)	86.0 (3.92)	136.2 (3.88)	136.2 (3.88)	136.2 (3.88)	136.2 (3.88)	136.2 (3.88)
2	81.0 (4.05)	81.0 (4.05)	81.0 (4.05)	81.0 (4.05)	81.0 (4.05)	129.0 (3.92)	**127.7** (3.98)	139.4 (4.04)	142.4 (4.05)	142.4 (4.05)
4	**70.1** (4.03)	72.9 (4.04)	73.6 (4.05)	73.6 (4.05)	73.6 (4.05)	124.6 (3.91)	114.5 (3.96)	**110.9** (4.02)	118.0 (4.08)	124.1 (4.10)
8	63.8 (4.02)	61.5 (4.01)	**61.1** (4.00)	62.2 (4.01)	62.3 (4.01)	123.6 (3.91)	108.4 (3.94)	96.6 (3.99)	**95.1** (4.03)	97.0 (4.06)
16	61.0 (4.02)	57.8 (4.00)	54.4 (3.97)	**53.7** (3.96)	54.3 (3.96)	121.1 (3.91)	104.6 (3.94)	87.9 (3.95)	79.8 (3.98)	**77.7** (3.99)

Table 6: **Commit vs. non-commit refinement-in-loop training on LM1B, over NFE $\times$ commit threshold κ.** The lowest Gen-PPL in each row of each arm is **bolded**. The on-policy rollout that produces the distillation target either hard-commits its confident tokens into the supervised state (*commit*) or re-maps the soft state without committing (*non-commit*), and both arms are trained for 200K steps with all other parameters held constant. The commit setting is used as default in ril training.

E Hyperparameter Discussion

Below, we discuss each ablated hyperparameter, what it controls, and the behavior that the sweeps in Table 10 show. App E.1 describes the training-time knobs of the map, and App E.2 describes the inference-time knobs that can be changed on a fixed checkpoint.

E.1 Training Hyperparameters

Adaptive Loss To stabilize training, we scale the losses with a gradient-detached weight that downweights large mismatches in the predicted and true target. Denoting the transport map prediction $q(x) = T(x) \in \Delta^{V-1}$ and $p(x) = \mathtt{sg}(T(x) + \dot{I}_t(x) \cdot \nabla T(x))$, we write the mismatch as $\Delta(x) = q(x) - p(x)$ and the weight as:

$$w(x) := \mathtt{sg}\left[\left(\|\Delta(x)\|_2^2 + c\right)^{-r}\right] \tag{167}$$

where c and r are tunable hyperparameters.

Minimum Anchor Time The threshold t_{anchor} gates the anchor loss, which supervises $T_\theta(I_t)$ directly toward x_1 only for $t > t_{\text{anchor}}$ and sets how much of the interpolant is treated as already committed. We prove in App. C.2 that the optimal anchor time aligns closely with the time of the phase transition and perform an ablation over t_{anchor} on LM1B in Table 12.

Noise Scale The prior scale σ in $\mu_0 = \mathcal{N}(0, \sigma^2 \boldsymbol{I}_V)^{\otimes L}$ sets the radius of the Gaussian hypersphere around the simplex, and enters the commitment time through the signal-to-noise ratio $\rho(t) = (1 - \alpha_t)/\alpha_t\sigma$ (See App C.2 and Figure 2). Raising it to σ=1.2 improves Gen-PPL at every NFE budget (178.1 vs. 247.1 at NFE 1, 61.9 vs. 70.1 at NFE 8) but drops entropy below 4 (Table 10).

Schedule Exponent The exponent a in $\alpha_t = (1-t)^a$ controls how quickly the interpolant resolves onto a vertex, and for larger a, this reduces the commitment time $t^\star$ (Figure 2). Setting a=1 reduces to the linear interpolant $I_t = (1-t)x_0 + tx_1$.

Adaptive Loss Constant The constant c in the adaptive weight $w(x) = \mathtt{sg}[(\|\Delta(x)\|_2^2 + c)^{-r}]$ determines how strongly a large prediction mismatch is downweighted, with small c giving a more aggressive reweighting. It is the least sensitive knob we swept, where moving c from 1 to 10^{-5} changes Gen-PPL negligibly and leaves entropy unchanged, demonstrating that our method is robust to minor tuning in the objective.

E.2 Inference Hyperparameters

Rounds of Refinement Each round is one map application and one NFE with no additional NFE from computing the confidence or quality score, so the number of rounds K is equivalent to the number of jumps taken with a flow map or integration steps taken with a flow matching model. Quality improves consistently with increasing k until the sampler saturates when every position is committed to a token.

Commit Scorer The scorer ranking positions for commitment is either the map's own softmax confidence (rounding error of argmax token) or the learned per-token quality q_ϕ. All positions with score larger than the confidence threshold κ are committed. The sampler commits a minimum of $n_r = \lceil |\mathcal{R}_r|/(k - r + 1) \rceil$ highest-scoring uncommitted positions at round regardless of whether their score exceeds κ, which guarantees the sampler finishes within its NFE budget.

Confidence Threshold The threshold κ decides which positions are committed rather than renoised, with position ℓ frozen when $q^\ell \geq \kappa$. Its effect depends on whether the per-round minimum commit count $n_r = \lceil |\mathcal{R}_r|/(k - r + 1) \rceil$ is met.

Commit Temperature Temperature used to scale the proposed categorical probabilities before taking the softmax $T_\theta(x_0) := \text{softmax}(f_\theta(x_0)/\mathtt{temp})$. When $\mathtt{temp} = 0.0$, the committed token is the argmax.

Prior Scale The standard deviation σ of the Gaussian prior distribution from which the initial noise latent at each position ℓ are sampled $x_0^\ell \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_V)$.

Repetition Penalty For unconditional language modeling, the commit rule reads each position's marginal independently within a round can cause repetition between tokens committed within a single round. As is standard for autoregressive decoding (Keskar et al., 2019), we apply a repetition penalty $\lambda \geq 0$ that divides $T_\theta(x)_j^\ell$ by $(1 + n_j(\mathcal{C}))^\lambda$ on uncommitted tokens before decoding and scoring, where $n_j(\mathcal{C})$ counts committed positions holding token j, which has no effect for 1 NFE sampling. We find that after refinement-in-loop training, the penalty is not required for coherent sampling, demonstrating that training on contexts seen during inference improves generation quality.

F Ablations and Additional Results

Overview We provide the results for hyperparameter ablations and additional experiments in Tables 10 to 14.

- Table 7 shows the Gen. PPL and entropy for the full sweep of NFEs up to 1024 on LM1B and OWT.
- Table 8 shows the accuracy for the full sweep of NFEs for TinyGSM/GSM8K against the state-of-the-art few-step baseline FMLM+.
- Table 9 compares integrating the autonomous field over increasing time horizons via the endpoint parameterization from Remark B.1 with applying the one-step map on LM1B.
- Table 10 shows hyperparameter ablations on LM1B which are used to inform the OWT experiments.
- Table 11 shows a sweep of NFE and commit threshold κ on each difficulty of Sudoku.
- Table 12 shows the results from sweeping the minimum anchor time $t_{\text{anchor}} \in \{0.60, 0.75, 0.80, 0.85\}$ during DBTM + ril training for LM1B.
- Table 13 shows the numerical results corresponding to the frontier curves in Figure 3 for LM1B, with the Gen-PPL at matched entropy on the Pareto frontiers generated from sweeping inference parameters for each method.
- Table 14 shows the numerical results corresponding to the frontier curves in Figure 1 for OWT, with the Gen-PPL at matched entropy on the Pareto frontiers generated from sweeping inference parameters for each method.

Integrating the Autonomous Flow We empirically determine the hitting time on LM1B by integrating the time-independent velocity implied by the trained map over a range of horizons H (Table 9). At short horizons ($H = 1$), the integrated trajectory does not reach the correct basin of attraction, but by the time the horizon increases to $H = 4$, the integrated field has settled onto a fixed point $x \in M_1$ where $b(x) = 0$, and sample quality no longer improves as the horizon grows to $H = 8$. However, we empirically observe that applying the one-step map $T_\theta(x)$ consistently outperforms integrating the implied velocity field, even with fine discretization steps ($N_{\text{steps}} = 64$). This supports scaling refinements of the one-step map rather than scaling integration steps, and is the property that motivates the refinement scheme of Section 5.

Dataset	Method	NFE 1	2	4	8	16	128	256	1024
LM1B	FMLM	119.34 (4.16)	110.19 (4.21)	98.76 (4.21)	86.28 (4.20)	78.06 (4.21)	57.73 (4.20)	51.03 (4.18)	41.84 (4.15)
	DFM (PSD)	94.08 (4.06)	87.42 (4.08)	78.89 (4.10)	69.90 (4.10)	64.90 (4.11)	58.38 (4.10)	56.59 (4.10)	56.31 (4.11)
	DFM (ESD)	68.11 (3.79)	77.60 (4.11)	71.53 (4.13)	65.61 (4.13)	59.92 (4.13)	55.70 (4.11)	56.88 (4.12)	58.27 (4.12)
	DBTM	201.20 (4.03)	167.05 (4.19)	119.99 (4.20)	90.19 (4.16)	71.42 (4.11)	58.81 (4.03)	58.81 (4.03)	58.81 (4.03)
	DBTM + ril	85.50 (3.92)	81.00 (4.05)	73.54 (4.04)	60.86 (4.00)	54.48 (3.97)	51.26 (3.95)	51.26 (3.95)	51.26 (3.95)
OWT	FMLM	168.30 (5.17)	133.29 (5.25)	111.31 (5.26)	88.27 (5.26)	68.06 (5.24)	28.73 (4.88)	20.26 (4.70)	9.10 (4.13)
	DFM (PSD)	180.29 (4.91)	152.83 (5.03)	122.32 (5.10)	98.54 (5.11)	82.51 (5.09)	56.00 (5.00)	51.81 (4.97)	47.82 (4.97)
	DFM (ESD)	5.33 (0.26)	108.91 (5.15)	77.08 (5.27)	62.98 (5.23)	55.03 (5.18)	41.90 (5.04)	39.08 (5.00)	36.48 (4.95)
	FMLM+	378.53 (5.34)	114.46 (5.14)	41.16 (4.77)	25.15 (4.53)	21.58 (4.42)	20.79 (4.40)	20.72 (4.39)	20.60 (4.37)
	DBTM (attn)	249.8 (4.77)	156.3 (5.06)	123.1 (5.18)	107.9 (5.20)	98.6 (5.20)	86.7 (5.20)	83.2 (5.20)	80.4 (5.22)
	DBTM + ril (attn)	133.0 (4.94)	109.6 (5.24)	115.5 (5.42)	120.0 (5.48)	121.0 (5.50)	120.5 (5.54)	117.6 (5.55)	117.6 (5.55)
	DBTM (lin)	188.2 (4.97)	76.2 (5.14)	52.2 (5.26)	42.2 (5.28)	37.0 (5.27)	31.6 (5.25)	30.7 (5.25)	30.1 (5.25)
	DBTM + ril (lin)	65.2 (4.95)	62.2 (5.38)	57.5 (5.52)	52.3 (5.56)	48.8 (5.57)	42.2 (5.57)	41.0 (5.56)	40.9 (5.56)

Table 7: **Full NFE sweep for LM1B and OWT.** Generative perplexity (GPT-2 Large; ↓) and entropy (↑) across NFEs for LM1B and OWT compared to few-step flow map baselines FMLM, discrete flow maps (DFM), and FMLM with posterior refinement (FMLM+). (attn) denotes the same DiT architecture as FMLM, and (lin) denotes the linear attention architecture with matched parameters described in App. G.2. FMLM and FMLM+ values are evaluated with published checkpoints, and DFM values are taken from the paper. All values are computed from 1024 sequences. For DBTM and FMLM+, NFE is the number of refinement rounds. DBTM LM1B checkpoints are evaluated at 200K steps, and DBTM OWT checkpoints are evaluated at 100K steps.

	GSM8K (NFE)						
Method	1	2	4	8	16	32	64
FMLM+	0	0.6	**5.7**	9.7	**14.5**	16.6	15.0
DBTM	**0.8**	**1.1**	5.7	**10.1**	14.3	**16.8**	**16.2**

Table 8: **GSM8K accuracy across NFE for DBTM vs. FMLM+.** Final-answer accuracy (%, ↑) on the full 1319-problem GSM8K test set, one greedy sample per question (k=1) from a single checkpoint per row, where NFE is the number of refinement rounds and both samplers spend exactly one network evaluation per round (**bold** = better of the two in the column). FMLM+ (Agarwal et al., 2026) is the published Posterior Refinement checkpoint with refinement threshold 0.99 and DBTM is the refinement-in-loop arm at q=0.8. Both methods are evaluated at 250K steps, commits ranked by softmax confidence, and fresh noise redrawn each round.

	Autonomous field (horizon H)			**One-step map**
	H=1	H=4	H=8	NFE 1
Gen-PPL ↓	3571.8	276.1	293.2	247.1
Entropy	4.822	4.378	4.386	4.115

Table 9: **Autonomous flow horizon sweep on LM1B.** Integrating the learned autonomous field b with 64 Euler steps up to horizon H, compared against a single application of the one-step map T_θ (DBTM without ril). At H=1, the flow has not converged and generations remain incoherent. At H=4, quality saturates, indicating convergence to the manifold. The one-step map outperforms the fully integrated field in one NFE, demonstrating the advantage of directly learning T.

Configuration	**One-step** (NFE 1) Gen-PPL ↓	Ent.	**Refinement** (NFE 8) Gen-PPL ↓	Ent.
Noise scale σ				
1.0	247.12	4.11	70.08	4.02
1.2	178.15	3.94	61.90	3.98
Schedule exponent a				
1	247.12	4.11	70.08	4.02
2	221.32	4.03	77.38	3.98
Adaptive-loss constant c				
1	247.12	4.11	70.08	4.02
10^{-5}	260.31	4.12	69.00	4.01
Commit scorer				
Softmax confidence	247.12	4.11	70.08	4.02
Learned q_ϕ (shared head)	247.12	4.11	65.31	3.75
Refinement rule				
Iterate the map $T_\theta^{\circ k}$	178.15	3.94	501.24	4.23
Renoise + remap	178.15	3.94	61.90	3.98

Table 10: **Ablation results on LM1B.** Each block varies one knob off the default parameters (clean-context per-token α, learned weighting, adaptive transport r=0.5, CE endpoint, OT coupling, linear interpolation, uniform time, batch 128) at t_{anchor}=0.75, σ=1.0, a=1, c=1; shaded = the setting the paper uses.

	Easy					**Medium**					**Hard**				
NFE (Rounds)	0.5	0.7	0.9	0.99	0.999	0.5	0.7	0.9	0.99	0.999	0.5	0.7	0.9	0.99	0.999
1	63.65	63.65	63.65	63.65	63.65	39.50	39.50	39.50	39.50	39.50	1.60	1.60	1.60	1.60	1.60
2	**94.40**	94.20	92.20	88.65	85.10	**85.75**	83.15	78.70	70.30	63.45	**17.70**	13.35	8.80	6.05	5.55
4	97.35	98.05	98.85	**99.70**	99.50	92.25	94.60	96.30	**97.65**	97.30	80.15	84.65	**86.05**	85.70	84.60
8	97.20	97.90	98.75	**99.80**	99.75	92.55	94.90	96.95	98.85	**99.15**	81.60	88.70	93.10	94.60	**95.00**
16	97.40	98.20	99.05	99.75	**99.90**	93.45	95.35	97.50	98.85	**99.35**	83.35	89.60	94.85	96.85	**97.45**

Table 11: **Commit threshold sweep for DBTM + ril on Sudoku.** Exact-solve accuracy (%) on the 2000-puzzle held-out set over NFE (refinement rounds) × commit threshold $\kappa \in \{0.5, 0.7, 0.9, 0.99, 0.999\}$ for the checkpoints reported in Table 2. Shaded = the fixed $\kappa = 0.999$ used there. The best in each row within each difficulty is **bolded**; NFE 1 is a one-shot decode and does not depend on κ. Lower thresholds commit more positions per round and plateau early, so at NFE 8-16 accuracy increases with κ on every difficulty, and $\kappa = 0.999$ is best or within one puzzle of best. At NFE 2 the ordering reverses, since a strict threshold leaves most positions to be forced by the commit floor in the final round, and at NFE 4 the optimum sits slightly below the strictest setting.

t_{anchor}	0.60				**0.75***			
NFE (Rounds)	0.7	0.9	0.99	0.999	0.7	0.9	0.99	0.999
1	47.4 (3.65)	47.4 (3.65)	47.4 (3.65)	47.4 (3.65)	92.9 (3.93)	92.9 (3.93)	92.9 (3.93)	92.9 (3.93)
2	41.3 (3.86)	41.3 (3.86)	41.3 (3.86)	41.3 (3.86)	88.0 (4.06)	88.0 (4.06)	88.0 (4.06)	88.0 (4.06)
4	35.8 (3.88)	36.1 (3.89)	36.1 (3.89)	36.1 (3.89)	79.5 (4.05)	81.0 (4.06)	81.0 (4.06)	81.0 (4.06)
8	29.1 (3.87)	29.7 (3.86)	29.9 (3.86)	29.9 (3.86)	66.5 (4.02)	66.5 (4.01)	68.3 (4.01)	68.3 (4.01)
16	27.1 (3.86)	26.0 (3.84)	25.9 (3.84)	25.9 (3.84)	62.4 (4.00)	58.4 (3.97)	59.1 (3.96)	59.4 (3.96)
t_{anchor}	0.80				0.85			
NFE (Rounds)	0.7	0.9	0.99	0.999	0.7	0.9	0.99	0.999
1	110.8 (3.97)	110.8 (3.97)	110.8 (3.97)	110.8 (3.97)	174.9 (4.12)	174.9 (4.12)	174.9 (4.12)	174.9 (4.12)
2	109.2 (4.09)	109.2 (4.09)	109.2 (4.09)	109.2 (4.09)	170.6 (4.23)	170.6 (4.23)	170.6 (4.23)	170.6 (4.23)
4	99.0 (4.09)	100.6 (4.10)	100.5 (4.09)	100.5 (4.09)	155.5 (4.23)	156.2 (4.23)	156.2 (4.23)	156.2 (4.23)
8	82.9 (4.06)	84.1 (4.04)	85.1 (4.05)	85.1 (4.05)	125.4 (4.18)	125.2 (4.17)	126.9 (4.18)	126.9 (4.18)
16	76.0 (4.04)	71.6 (4.00)	72.4 (3.99)	72.7 (4.00)	112.6 (4.16)	103.5 (4.12)	105.4 (4.12)	105.2 (4.12)

Table 12: **Effect of the minimum anchor time t_{anchor} for DBTM + ril training on LM1B.** Gen-PPL (entropy) over NFE (refinement rounds) $\times$ threshold grid across $t_{\text{anchor}} \in \{0.60, 0.75, 0.80, 0.85\}$, where * indicates the default $t_{\text{anchor}} = 0.75$ for LM1B and OWT. Columns within each anchor are commit thresholds ($\kappa \in \{0.7, 0.9, 0.99, 0.999\}$) and rows are NFE (refinement rounds); the $\kappa = 0.9$ columns used in the main results are highlighted. Thresholds maintain constant performance at low rounds and take effect only at rounds 8–16, where stricter thresholds lower Gen-PPL.

		Best			**Gen-PPL at matched entropy**						
NFE	Method	Gen-PPL ↓	at Ent.	Ent. range	3.90	4.00	4.10	4.15	4.20	4.25	4.30
1	FMLM	112.8	4.124	4.12–4.18	–	–	–	**116.2**	–	–	–
	DBTM	199.5	4.045	4.04–4.63	–	–	252.0	370.2	459.3	557.3	685.9
	DBTM + ril	**86.0**	3.917	3.92–4.40	–	**122.4**	**210.5**	242.2	**306.2**	**388.1**	**497.7**
2	FMLM	104.6	4.196	4.20–4.32	–	–	–	–	**105.2**	**122.9**	**139.4**
	DBTM	147.0	4.084	4.08–4.22	–	–	147.7	156.7	177.5	–	–
	DBTM + ril	**71.6**	3.967	3.97–4.25	–	**75.1**	**89.1**	**99.3**	112.0	128.5	–
4	FMLM	93.4	4.207	4.21–4.32	–	–	–	–	–	109.0	124.8
	DBTM	101.4	4.046	4.05–4.32	–	–	105.7	115.5	140.7	151.5	164.1
	DBTM + ril	**62.5**	3.926	3.93–4.40	–	**69.7**	**79.8**	**86.4**	**93.8**	**103.0**	**113.6**
8	FMLM	85.3	4.190	4.19–4.30	–	–	–	–	85.4	98.8	–
	DBTM	74.8	3.986	3.99–4.31	–	75.5	85.8	93.2	109.5	117.1	126.2
	DBTM + ril	**49.5**	3.867	3.87–4.45	**50.8**	**60.1**	**68.6**	**73.2**	**78.7**	**85.4**	**92.7**
16	FMLM	76.7	4.219	4.22–4.31	–	–	–	–	–	89.6	115.7
	DBTM	60.3	3.910	3.91–4.28	–	65.2	76.0	85.4	90.5	96.1	–
	DBTM + ril	**44.2**	3.832	3.83–4.47	**46.6**	**57.3**	**62.1**	**66.1**	**70.4**	**75.8**	**81.7**

Table 13: **LM1B decode frontier shown in Figure 3: DBTM vs. FMLM, swept over several inference-time knobs.** Each method is swept over the knobs that trace its own coherence-diversity trade-off, all of its sweeps are pooled, and the Pareto frontier is taken (a point survives if no other has both higher entropy and lower Gen-PPL). The best in the column within each NFE block are **bolded**. Blank cells indicate entropy values where the method produces no point on the Pareto frontier. **Swept parameters.** *DBTM* (both rows): (i) **commit temperature** `temp` $\in [0, 1.2]$, (ii) **prior scale** $\sigma \in [0.6, 5]$, (iii) **commit threshold** $\kappa \in [0.1, 0.999]$, (iv) **repetition penalty** $\lambda \in \{0, 0.5, 1\}$. *FMLM*: (i) **churn** $\gamma \in [0, 0.99]$, the noise re-injected between sampler steps, and (ii) the same **prior scale** σ, applied to its initial latent.

NFE	Method	Best			Gen-PPL at matched entropy									
		Gen-PPL ↓	at Ent.	Ent. range	5.00	5.10	5.20	5.30	5.35	5.40	5.45	5.50	5.60	5.70
1	FMLM	168.2	5.174	5.17–5.40	–	–	177.9	249.0	306.0	372.4	–	–	–	–
	FMLM+	378.5	5.344	5.34–5.50	–	–	–	–	384.4	433.9	509.4	–	–	–
	DBTM (attn)	369.9	5.001	4.90–5.16	369.3	457.1	–	–	–	–	–	–	–	–
	DBTM + ril (attn)	253.3	5.171	5.17–5.54	–	–	274.9	361.6	423.6	485.6	547.6	609.6	–	–
	DBTM (lin)	234.0	5.102	4.97–5.31	196.4	233.3	291.0	366.1	–	–	–	–	–	–
	DBTM + ril (lin)	**113.6**	5.098	4.95–5.57	**82.4**	**114.5**	**161.7**	**225.6**	**276.5**	**327.4**	**378.3**	**429.2**	–	–
2	FMLM	137.4	5.265	5.26–5.36	–	–	–	163.2	235.2	–	–	–	–	–
	FMLM+	114.5	5.136	5.14–5.60	–	–	142.6	223.5	299.6	379.5	432.4	515.6	666.7	–
	DBTM (attn)	224.2	5.075	4.96–5.47	194.2	227.4	240.3	270.6	304.3	348.1	394.5	–	–	–
	DBTM + ril (attn)	199.5	5.370	5.37–5.78	–	–	–	–	–	221.5	246.5	290.8	393.1	466.8
	DBTM (lin)	76.6	5.143	4.85–5.77	**66.8**	**73.6**	90.7	116.5	136.2	159.3	186.1	238.3	392.7	454.3
	DBTM + ril (lin)	**43.3**	5.113	5.11–5.94	–	–	**49.8**	**53.8**	**58.6**	**69.2**	**85.1**	**92.3**	**141.5**	**250.9**
4	FMLM	119.1	5.296	5.30–5.36	–	–	–	121.6	185.5	–	–	–	–	–
	FMLM+	111.4	5.165	4.80–5.63	87.1	101.8	126.8	218.4	245.2	272.0	298.8	325.6	464.5	–
	DBTM (attn)	168.0	5.016	4.98–5.65	152.3	169.1	176.9	184.8	194.4	224.4	252.3	304.4	379.9	–
	DBTM + ril (attn)	169.2	5.376	5.38–5.97	–	–	–	–	–	170.7	173.7	176.6	195.3	227.4
	DBTM (lin)	47.5	5.236	4.83–5.97	**40.9**	**43.7**	46.5	60.3	72.5	82.4	96.0	139.2	170.9	274.0
	DBTM + ril (lin)	**34.7**	5.128	5.13–6.03	–	–	**39.2**	**42.0**	**42.4**	**44.5**	**48.0**	**51.9**	**77.2**	**90.3**
8	FMLM	95.5	5.280	5.28–5.38	–	–	–	99.9	131.0	–	–	–	–	–
	FMLM+	69.0	5.011	4.82–5.66	67.0	87.4	99.2	147.1	160.4	189.5	229.2	261.3	318.1	–
	DBTM (attn)	137.9	5.072	4.97–5.72	118.9	139.8	146.7	153.5	163.9	187.9	197.7	233.1	291.5	448.6
	DBTM + ril (attn)	166.4	5.645	5.64–6.10	–	–	–	–	–	–	–	–	–	180.6
	DBTM (lin)	**33.3**	5.269	4.88–6.03	**26.4**	**28.9**	**31.5**	43.6	46.7	51.6	61.3	74.1	167.7	186.5
	DBTM + ril (lin)	37.2	5.423	4.82–6.04	32.8	33.9	34.9	**35.9**	**36.4**	**37.0**	**38.8**	**41.9**	**52.6**	**71.8**
16	FMLM	76.8	5.267	5.27–5.44	–	–	–	84.8	88.7	95.6	–	–	–	–
	FMLM+	62.0	5.001	4.96–5.63	61.7	75.5	89.1	140.6	143.2	145.9	202.7	251.5	295.1	–
	DBTM (attn)	120.7	5.047	4.94–5.81	110.0	123.9	130.1	136.2	141.1	148.3	155.6	184.7	257.7	402.3
	DBTM + ril (attn)	131.3	5.614	5.61–6.11	–	–	–	–	–	–	–	–	–	135.6
	DBTM (lin)	**21.3**	5.005	4.88–5.99	**21.2**	**23.5**	**25.8**	**37.5**	**37.6**	41.6	54.3	63.5	94.1	141.9
	DBTM + ril (lin)	30.1	5.355	5.36–6.12	–	–	–	–	–	**32.0**	**34.2**	**36.2**	**42.3**	**48.9**

Table 14: **OWT decode frontier shown in Figure 1: DBTM vs. FMLM and FMLM+, swept over several inference-time knobs.** Companion to the decode-frontier figure. Each method is swept over the knobs that trace its own coherence-diversity trade-off; all of its sweeps are pooled, and the Pareto frontier is taken (a point survives if no other has both higher entropy and lower Gen-PPL). The best in the column within each NFE block are **bolded**. Blank cells indicate entropy values where the method produces no point on the Pareto frontier. Entropy is the within-sequence token-frequency entropy, with real OWT at entropy 5.441. (attn) denotes the same DiT architecture as FMLM, and (lin) denotes the linear attention architecture with matched parameters described in App. G.2. FMLM and FMLM+ values are evaluated with published checkpoints. **Swept parameters.** *DBTM*: (i) **commit temperature** $\texttt{temp} \in \{0, 0.7, 1.0\}$, (ii) **prior scale** $\sigma \in [0.6, 5]$, (iii) **commit threshold** $\kappa \in [0.1, 0.999]$, (iv) **repetition penalty** $\lambda \in \{0, 0.5, 1\}$. *FMLM*: (i) **churn** $\gamma \in \{0, 0.1, 0.2, 0.3, 0.5\}$, the noise re-injected between sampler steps, and (ii) a **commit temperature** $\texttt{temp} \in \{0, 0.5, 0.7, 0.85, 1.0\}$ at its final argmax, which was added on top of the method. *FMLM+*: (i) the **split of the NFE budget** into refinement rounds $\times$ ODE substeps, (ii) the **commit schedule**, a uniform per-round budget or a fixed top-$k \in \{64, 128, 256, 384, 768\}$ per round, (iii) **commit threshold** $\kappa \in \{0.9, 0.999\}$, and (iv) a **commit temperature** $\texttt{temp} \in [0, 2]$ at its decode, also added on top of the method.

G Experiment Details

G.1 General Experiment Details

Training Details All models use AdamW (β_1=0.9, β_2=0.999, ϵ=10^{-8}) in bf16, zero weight decay, gradient clipping at 1.0, and peak learning rate 3×10^{-4} with linear warmup. We maintain an exponential moving average (EMA) of the map weights with decay 0.9999, and a second, independent EMA of the predictor ϕ with the same decay. All reported results use the EMA weights. The global batch is given by per-GPU batch $\times$ devices $\times$ accumulation, so runs reproduce at any device count: 512 (per-GPU 128) for language modeling, 256 for reasoning.

Architecture The backbone architecture is a DiT (Peebles and Xie, 2023) with adaLN conditioning (dimension 128) and rotary positional embeddings (Su et al., 2024), acting on a continuous (B, L, V) simplex latent without conditioning on time t. Language modeling and TinyGSM use 768 hidden / 12 layers / 12 heads, Sudoku 512 / 8 / 8, all with FFN multiplier 4, dropout 0.1, untied embeddings, and a tanh softcap of 50 on attention logits (Table 15).

Fused Kernel for JVP Computation The transport loss (14) is defined through $\dot{I}_t \cdot \nabla T_\theta(I_t)$, so each step needs a Jacobian-vector product (JVP) through the network, evaluated in forward mode. This dominates our training cost since it requires materializing the `(B, heads, L, L)` score tensor. We instead use a fused Triton kernel (Zhou et al., 2026) that computes the attention output and its derivative together in a single streaming pass, so the full score tensor is never held in memory. The two passes share one sweep over the keys and values: the ordinary forward pass only reserves the output buffers, and the kernel fills them while computing the derivative.

Backpropagation is handled by two separate kernels, one for the output and the derivative, which can be cleanly switched off to the standard JVP mechanism by setting `fast_jvp_attn=false`. Measured on an A100 over a four-block DiT, it cuts peak memory by 1.4× at B=64, L=128, V=12 and by 3.7× at B=8, L=1024, V=8192, with a 1.2× and 1.7× speedup respectively, while matching the unfused implementation to a cosine similarity of at least 0.99999 on the prediction, the derivative, and the gradients. One limitation is that the kernel does not implement the tanh softcap on attention logits. We therefore checked on each task that the logits stay well below the cap before using the fused kernel.

G.2 Unconditional Language Modeling Experiment Details

Datasets and Tokenization LM1B uses the `bert-base-uncased` tokenizer (V=30,522) at length 128 and OpenWebText uses GPT-2 BPE (V=50,257) at length 1024. Documents are concatenated and wrapped into fixed-length blocks with EOS at every boundary, so all sequences are full and no padding is needed.

Linear-Attention Backbone The **DBTM + lin** rows replace nine of the twelve attention blocks with bidirectional gated delta-rule mixers following DeltaFlow (Guo et al., 2026), keeping full attention at every fourth block (`[gdn, gdn, gdn, attn]`, chunk size 256, alternating scan direction). Each gated DeltaNet (Yang et al., 2025; 2024) runs forward and backward over the sequence. Since DBTM is autonomous, the decay and write-rate gates that DeltaFlow conditions on diffusion time are instead conditioned on the current simplex state. We do not use DeltaFlow's temporal-state-consistency loss. Hidden size, depth, heads, dropout, the predictor ϕ and every training knob are identical to the DiT architecture with matched parameter count (171.9M for attention, 177.5M for linear), so the two backbones differ only in the attention mechanism. The linear rows are trained from scratch at the same batch size and learning rate and reported at the same step count. The delta-rule attention mechanism costs time linear in the sequence length L, whereas dense attention costs quadratic. In Figure 1, Table 1, and Table 7, we show the linear architecture on OpenWebText (L=1024) as a proof-of-concept for long-context modeling.

Training Configuration Both datasets use a global batch of 512 and 10,000 warmup steps, a linear interpolant with σ=1.0, t_{anchor}=0.75, OT coupling, uniform time sampling, adaptive transport and cross-entropy endpoint losses, and the `clean_context` per-token schedule, which trains the map on partially committed sequences and so matches the refinement sampler.

Baselines For unconditional language modeling on LM1B and OWT, we compare DBTM against distilled discrete diffusion methods, including Duo (Sahoo et al., 2025) with DCD (Sahoo et al., 2025), MDLM (Sahoo et al., 2024) with SDTT (Deschenaux and Gulcehre, 2025), and both Duo and

MDLM with Di4C (Hayakawa et al., 2025), and flow map methods, including categorical flow maps (CFM) (Roos et al., 2026), flow map language models (FMLM) (Lee et al., 2026a), FMLM with posterior refinement (FMLM+) (Agarwal et al., 2026), and discrete flow maps (DFM) with both the semigroup (PSD) and Eulerian (ESD) objectives (Potaptchik et al., 2026). We evaluate FMLM+ using the released checkpoints with one integration step per refinement round, like DBTM. Baseline values are taken from Lee et al. (2026a); Potaptchik et al. (2026) or evaluated from published checkpoints.

Evaluation Metrics We report generative perplexity (Gen. PPL) scored by GPT-2 Large (Radford et al., 2019) and sample entropy over 1024 generated sequences. The goal is to minimize Gen. PPL while maximizing entropy to match the data entropy of LM1B (`entropy`=4.336) and OWT (`entropy`=5.441).

G.3 Reasoning Experiment Details

Datasets and Tokenization Following S-FLM (Deschenaux and Gulcehre, 2026) and FMLM+ (Agarwal et al., 2026), Sudoku puzzles with unique solutions are generated at three difficulties set by clue count (easy 40, medium 35, hard 30), with 48,000 train and 2,000 held-out puzzles per difficulty. Sudoku uses a 12-token vocabulary (empty, digits 1-9, row separator, BOS) and a grid is 81 cells plus 8 separators = 89 tokens, giving the layout `[BOS] puzzle(89) [BOS] solution(89)` = 180 tokens with prompt length 91 and loss on the solution half. TinyGSM (Liu et al., 2023a) pairs grade-school word problems with Python programs, tokenized by SmolLM-135M (V=49,152) as `[BOS] question \n program [EOS]` at length 512, supervising the program region only. Evaluation uses the real GSM8K test set (1,319 problems) that was never trained on.

Training Configuration Sudoku uses the 512-wide backbone (≈30M parameters) at L=180 and TinyGSM the 768-wide one (≈165M) at L=512, with global batch 256 and 30,000 / 50,000 warmup steps respectively. Transport-map training matches the language setting (linear interpolant, σ=1.0, OT coupling, uniform time, `clean_context`, adaptive transport and cross-entropy endpoint losses, anchor and boundary weights 1.0), with t_{anchor}=0.69 for Sudoku and t_{anchor}=0.82 for TinyGSM. The predictor ϕ is a 256-wide, 4-layer, 4-head head with quality weight 1.0, positive-class reweighting, clean-context labels, and BCE.

Baselines For the reasoning tasks, we compare DBTM against many-step autoregressive sampling (AR), discrete diffusion methods, including MDLM (Sahoo et al., 2024) and Duo (Sahoo et al., 2025), continuous flow methods, including CANDI (Pynadath et al., 2026a), FLM (Lee et al., 2026a), and S-FLM (Deschenaux and Gulcehre, 2026), and the few-step flow map baseline FMLM+ (Agarwal et al., 2026). Baseline values are taken from Agarwal et al. (2026) or evaluated from published checkpoints.

Evaluation Metrics For Sudoku, we report exact-solve accuracy over the 2,000 held-out puzzles. For GSM8K, we report answer accuracy by executing the generated program in a sandbox against the true answer, counting parse or execution failures as incorrect. All numbers use EMA weights and argmax commitment.

	LM1B	OpenWebText	TinyGSM / GSM8K	Sudoku
Architecture (DiT)				
Hidden size	768	768	768	512
adaLN cond. dim	128	128	128	128
Layers	12	12	12	8
Attention heads	12	12	12	8
FFN multiplier	4	4	4	4
Dropout	0.1	0.1	0.1	0.1
Sequence length L	128	1024	512	180
Vocabulary $\|V\|$	30,522	50,257	49,152	12
Parameters (approx.)	$\sim$140M	$\sim$170M	$\sim$165M	$\sim$30M
Optimization				
Optimizer	AdamW (β_1=0.9, β_2=0.999, ϵ=10^{-8})			
Weight decay	0			
Gradient clipping	1.0			
Schedule	costant LR with linear warmup			
Peak LR	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Max steps	250,000	150,000	250,000	200,000
Global batch size	512	512	512	256
Per-GPU batch size	128	128	32	64
Precision	bf16			
EMA decay	0.9999			
Transport map training				
Interpolant schedule	linear	linear	linear	linear
Noise scale σ	1.0	1.0	1.0	1.0
OT coupling	on			
Per-token schedule	clean_context			
Time schedule	uniform			
Minimum anchor time t_{anchor}	0.75	0.75	0.82	0.69
Anchor λ / semigroup λ / boundary λ	1.0 / 0 / 1.0	1.0 / 0 / 1.0	1.0 / 0 / 1.0	1.0 / 0 / 1.0
Adaptive stabilizer c (transport / endpoint)	1.0	1.0	1.0	1.0
Adaptive weight r (transport / endpoint)	0.5 / 0.5	0.5 / 0.5	0.5 / 0.5	0.5 / 0.5
Inference hyperparameters				
Commit threshold κ	0.9	0.3	0.8	0.999
Commit scorer	softmax conf.	softmax conf.	softmax conf.	μ (ϕ head)
Commit temperature	0 (argmax)	0 (argmax)	0 (argmax)	0 (argmax)
Prior scale σ	1.0	1.0	1.0	1.0
Repetition penalty λ	1.0	1.0	0	0

Table 15: **Per-task default hyperparameters.** For model architectures and hyperparameters shared with baselines, we mostly follow the parameters used in the FMLM (Lee et al., 2026a) baseline for LM1B and OWT, and the FMLM+ (Agarwal et al., 2026) baseline for Sudoku and TinyGSM/GSM8K. Note that the base hyperparameters in App E may be different across ablations of different parameters but are constant within each parameter.

H ALGORITHMS

Algorithm 1 Training Discrete Beckmann Transport Map

Require: Samples $\{x_0\} \sim \mu_0$, $\{x_1\} \sim \mu_1$; interpolant α_t; transport map T_θ; loss weights $\lambda_{\text{bnd}}, \lambda_{\text{semi}}, \lambda_{\text{anchor}}$; anchor minimum time t_{anchor}; self-composition depths $\mathcal{K}$; learning rate η

while not converged **do**
  Sample a minibatch $\mathcal{B}$; reset all losses to 0
  for $(x_0, x_1) \in \mathcal{B}$ **do**
    Sample $t \sim \mathcal{U}(0,1)$
    $I_t^\ell \leftarrow \alpha_t x_0^\ell + (1-\alpha_t) x_1^\ell, \qquad \dot{I}_t^\ell \leftarrow \dot{\alpha}_t\,(x_0^\ell - x_1^\ell)$ for all ℓ
    $\mathcal{L}_{\text{transport}} \leftarrow \mathcal{L}_{\text{transport}} + \sum_\ell \left|T_\theta(I_t^\ell) - \texttt{sg}\big(T_\theta(I_t^\ell) + \dot{I}_t^\ell \cdot \nabla T_\theta(I_t^\ell)\big)\right|^2$ ▷ *transport*
    $\mathcal{L}_{\text{bnd}} \leftarrow \mathcal{L}_{\text{bnd}} + \sum_\ell \text{CE}\big(T_\theta(x_1)^\ell,\, x_1^\ell\big)$ ▷ *boundary*
    $\mathcal{L}_{\text{anchor}} \leftarrow \mathcal{L}_{\text{anchor}} + \sum_\ell \mathbf{1}[t > t_{\text{anchor}}]\, \text{CE}\big(T_\theta(I_t)^\ell,\, x_1^\ell\big)$ ▷ *anchor near-clean tokens*
    $\mathcal{L}_{\text{semi}} \leftarrow \mathcal{L}_{\text{semi}} + \sum_{k\in\mathcal{K}} \sum_\ell \text{CE}\big(T_\theta(I_t)^\ell,\, \texttt{sg}\big(T_\theta^{\circ k}(I_t)\big)^\ell\big)$ ▷ *self-distill toward k-fold refinement*
  end for
  $\mathcal{L}_{\text{total}} \leftarrow \frac{1}{|\mathcal{B}|}\Big[\mathcal{L}_{\text{transport}} + \lambda_{\text{bnd}}\mathcal{L}_{\text{bnd}} + \lambda_{\text{anchor}}\mathcal{L}_{\text{anchor}} + \lambda_{\text{semi}}\mathcal{L}_{\text{semi}}\Big]$ ▷ *total loss*
  $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}_{\text{total}}$
end while
return T_θ

Algorithm 2 Sampling Discrete Beckmann Transport Map with Refinement (ROLLOUT)

Require: Trained autonomous transport map $T_\theta : \mathbb{R}^{L\times V} \to \mathbb{R}^{L\times V}$; confidence threshold κ; refinement depth k; prior noise level σ; refinement mode $\in \{\texttt{quality}, \texttt{confidence}, \texttt{none}\}$; commit mode $\in \{\texttt{commit}, \texttt{non-commit}\}$

$\hat{x}_{0,k} \leftarrow (x_0^1, \ldots, x_0^L)$, $x_0^\ell \sim \mathcal{N}(\mathbf{0}, \sigma \boldsymbol{I}_V)$; $\texttt{trace} \leftarrow [\,]$ ▷ *every token starts as prior noise*
for $r = 1, \ldots, k$ **do**
  $(\hat{h}, \hat{x}) \leftarrow T_\theta(\hat{x}_{r-1,k})$ ▷ *hidden representation $\hat{h}$ and categorical proposal $\hat{x}$*
  $n_r \leftarrow \big\lceil |\mathcal{R}_r|/(k-r+1) \big\rceil$; $\mathcal{C}_r \leftarrow \emptyset$ ▷ *commit floor for round r*
  for $\ell = 1, \ldots, L$ **do**
    if `quality` **then** $q_\phi^\ell(\hat{x}) \leftarrow \text{QUALITYHEAD}(\hat{h})$ ▷ *learned token quality*
    else if `confidence` **then** $q_\phi^\ell(\hat{x}) \leftarrow \langle \arg\max(\hat{x}^\ell),\, \hat{x}^\ell \rangle$ ▷ *rounding confidence*
    else $q_\phi^\ell(\hat{x}) \leftarrow 1$
    end if
    if $q_\phi^\ell(\hat{x}) < \kappa$ **and** $|\mathcal{C}_r| > n_r$ **then**
      $\hat{x}_{r,k}^\ell \sim \mathcal{N}(\mathbf{0}, \sigma \boldsymbol{I}_V)$ ▷ *low quality and floor already met: fresh noise*
    else if `commit` **then**
      $\hat{x}_{r,k}^\ell \leftarrow x_1^\ell$, $\mathcal{C}_r \leftarrow \mathcal{C}_r \cup \{\ell\}$ ▷ *commit hard token*
    else if `non-commit` **then**
      $\hat{x}_{r,k}^\ell \leftarrow \hat{x}^\ell$, $\mathcal{C}_r \leftarrow \mathcal{C}_r \cup \{\ell\}$ ▷ *hold soft token distribution*
    end if
  end for
  $\texttt{trace.append}\big(\hat{x}_{r,k}, \mathcal{C}_r\big)$ ▷ *record partially committed sequence*
end for
return $\hat{x}_{k,k}$, $\texttt{trace} = \{(\hat{x}_{r,k}, \mathcal{C}_r)\}_{r=1}^k$

Algorithm 3 Refinement-in-Loop Losses (RIL)

Require: Sample pair (x_0, x_1); transport map T_θ; quality head q_ϕ; refinement depth k; current `step`; `warmup`; flag `quality`

$\mathcal{L}_{\text{ril-c}}, \mathcal{L}_{\text{semi-r}}, \mathcal{L}_{\text{quality}} \leftarrow 0$
$\{(\hat{x}_{r,k}, \mathcal{C}_r)\}_{r=1}^{k} \leftarrow \textsc{Rollout}(x_0, x_1, k)$ ▷ *Algorithm 2*
for $r = 1, \dots, k$ **do**
  $\mathcal{L}_{\text{ril-c}} \leftarrow \mathcal{L}_{\text{ril-c}} + \sum_{\ell \notin \mathcal{C}_r} \text{CE}\big(T_\theta(\hat{x}_{r-1,k})^\ell,\ x_1^\ell\big)$ ▷ *supervise uncommitted tokens*
  if `quality` **then**
    $\hat{x}_1^\ell \leftarrow \arg\max T_\theta(\hat{x}_{r-1,k})^\ell$ for all ℓ
    $\mathcal{L}_{\text{quality}} \leftarrow \mathcal{L}_{\text{quality}} + \sum_\ell \text{BCE}\big(\mathbf{1}[\hat{x}_1^\ell = x_1^\ell],\ q_\phi^\ell(T_\theta(\hat{x}_{r-1,k}))\big)$
  end if
  if `step` $>$ `warmup` **then**
    $\mathcal{L}_{\text{semi-r}} \leftarrow \mathcal{L}_{\text{semi-r}} + \sum_{\ell \notin \mathcal{C}_r} \text{CE}\big(T_\theta(\hat{x}_{r-1,k})^\ell,\ \texttt{sg}\big(T_\theta(\hat{x}_{k-1,k})\big)^\ell\big)$ ▷ *self-distill toward final round*
  end if
end for
return $\mathcal{L}_{\text{ril-c}},\ \mathcal{L}_{\text{semi-r}},\ \mathcal{L}_{\text{quality}}$

Algorithm 4 Training Discrete Beckmann Transport Map with Partial Context

Require: Samples $\{x_0\} \sim \mu_0$, $\{x_1\} \sim \mu_1$; interpolant α_t; transport map T_θ; quality head q_ϕ; loss weights $\lambda_{\text{bnd}}, \lambda_{\text{anchor}}, \lambda_{\text{semi}}, \lambda_{\text{ril}}$; anchor minimum time t_{anchor}; `warmup`; `max_steps`; learning rate η; flags `quality`, `ril`

for `step` $= 1, \dots,$ `max_steps` **do**
  Sample a minibatch $\mathcal{B}$; reset all losses to 0
  for $(x_0, x_1) \in \mathcal{B}$ **do**
    Sample clean fraction $f \sim \mathcal{U}(0,1)$ and noisy time $t \sim \mathcal{U}(0,1)$
    for $\ell = 1, \dots, L$ **do**
      $c_\ell \sim \text{Bernoulli}(f)$; $t_\ell \leftarrow c_\ell + (1 - c_\ell)\, t$ ▷ $c_\ell = 1$*: token ℓ is clean context*
      $I_t^\ell \leftarrow \alpha_{t_\ell} x_0^\ell + (1 - \alpha_{t_\ell}) x_1^\ell, \qquad \dot{I}_t^\ell \leftarrow \dot{\alpha}_{t_\ell}\,(x_0^\ell - x_1^\ell)$
    end for
    $\mathcal{L}_{\text{transport}} \leftarrow \mathcal{L}_{\text{transport}} + \sum_\ell |T_\theta(I_t^\ell) - \texttt{sg}\big(T_\theta(I_t^\ell) + \dot{I}_t^\ell \cdot \nabla T_\theta(I_t^\ell)\big)|^2$ ▷ *transport*
    $\mathcal{L}_{\text{bnd}} \leftarrow \mathcal{L}_{\text{bnd}} + \sum_\ell \text{CE}\big(T_\theta(x_1)^\ell,\ x_1^\ell\big)$ ▷ *boundary*
    $\mathcal{L}_{\text{anchor}} \leftarrow \mathcal{L}_{\text{anchor}} + \sum_\ell \mathbf{1}[t_\ell > t_{\text{anchor}}]\ \text{CE}\big(T_\theta(I_t)^\ell,\ x_1^\ell\big)$ ▷ *anchor near-clean tokens*
    if `quality` **then**
      $\hat{x}_1^\ell \leftarrow \arg\max T_\theta(I_t)^\ell$ for all ℓ
      $\mathcal{L}_{\text{quality}} \leftarrow \mathcal{L}_{\text{quality}} + \sum_\ell \text{BCE}\big(\mathbf{1}[\hat{x}_1^\ell = x_1^\ell],\ q_\phi^\ell(T_\theta(I_t))\big)$ ▷ *predict per-token correctness*
    end if
    if `ril` **then**
      $(\ell_{\text{ril-c}}, \ell_{\text{semi-r}}, \ell_{\text{quality}}) \leftarrow \text{RIL}(x_0, x_1, k, \texttt{step})$ ▷ *Algorithm 3*
      $\mathcal{L}_{\text{ril-c}} \leftarrow \mathcal{L}_{\text{ril-c}} + \ell_{\text{ril-c}}$; $\mathcal{L}_{\text{semi-r}} \leftarrow \mathcal{L}_{\text{semi-r}} + \ell_{\text{semi-r}}$; $\mathcal{L}_{\text{quality}} \leftarrow \mathcal{L}_{\text{quality}} + \ell_{\text{quality}}$
    end if
  end for
  $\mathcal{L}_{\text{total}} \leftarrow \frac{1}{|\mathcal{B}|}\Big[\mathcal{L}_{\text{transport}} + \lambda_{\text{bnd}}\mathcal{L}_{\text{bnd}} + \lambda_{\text{anchor}}\mathcal{L}_{\text{anchor}} + \lambda_{\text{ril}}\mathcal{L}_{\text{ril-c}} + \lambda_{\text{semi}}\mathcal{L}_{\text{semi-r}}\Big]$ ▷ *total loss*
  $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}_{\text{total}}$; $\phi \leftarrow \phi - \eta\, \nabla_\phi \mathcal{L}_{\text{quality}}$
end for
return T_θ, q_ϕ

I Example Generations

Figure 8 shows unconditional LM1B samples drawn under budgets of 1, 2, 4, and 8 function evaluations, each box annotated with the generative perplexity and entropy of that sample. Figures 12 and 13 show two TinyGSM solutions across a 32-round refinement trajectory. The conditioning question is set in gray and stays fixed, while generated positions move from purple to turquoise. Figure 14 applies the same coloring over to a Sudoku-Hard puzzle solved in 4 NFEs.

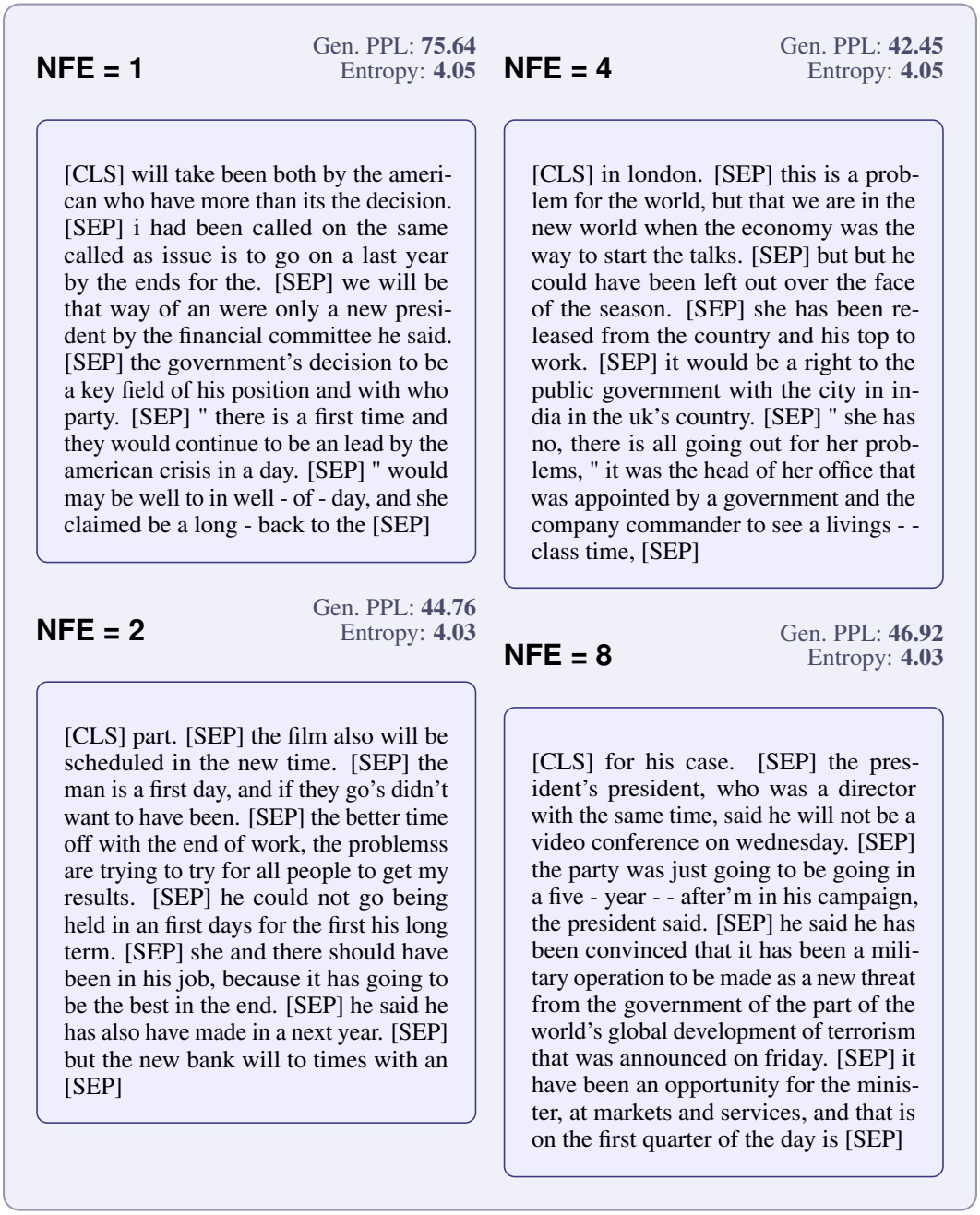

Figure 8: **Example LM1B generations from DBTM at 1, 2, 4, and 8 NFEs** (each NFE is one refinement round). Generative perplexity and entropy are reported per length-128 sample.

NFE = 16

Gen. PPL: **45.5**
Entropy: **5.58**

[END]...I mean, I think we started at the end of last year to figure out what we're doingre doing right now. And mean… when you look at your [data], you know that about $100,000 per year — That sounds pretty hard; i'm still sure little bit." But I think this is going to increase increased by about $100,000 per year — That's pretty single very important! This is very important and it's 'sthe single most important thing in our history. And so far i've've seen huge number million Americans coming across every day months' past decade...Our population has increased by about $100,000 per year — those people who don't caretakers." This is consistent with the United States — basically what's called "the United Kingdom" which translates into about $100,000 per year — What happens happening right? AndIt's called "the United States Of America" which " says "You'll say that this country is worth less than $100,000 annually,000-$100 per year, and then we'll be able access through some sort of so-called "takers." But at exact same time when we comes places where we're going to be able to make those kinds, we can make those kinds right now, but I don't becauset really think there is no best way to do something We don't becauset know what it is important for us.

I think it's really very important because I think it's important for us to do something. The interesting thing about'vesurveillance since America is that when living love this country we are going to come up with something like this: What [What_surveillance?" There—A lot those people living love the United States] [that wasnt true]. They thought they were going to do something like — even though they were going would have enough opportunity to do something like this.)

For many countries around the world, the problem is probably maybe two or two years ago when somebody thought they're going to do something like "surve surveillance." But I think there is going to be a lot amount of work that needs to happen on the other side of things, along with things that needs to be done now So We're going to put some factors into account exactly where we need irreveillanceillance, so that everybody cansolve themselves around each issues in terms of how much work should spend, obviously, if you look doing something, you need to focus on the other side of things that You need can deal with them! For So, we need to focus on… You'll need to focus on making decisions while spending whatever whatever needs will happen in the future, so we need focus on making decisions decisions and these kinds of things that we can actually happen in foreseeable future.

The second between here is as well as how much money should actually spend, especially at the rest of our lives. So, because many people on the other side of things are trying to get rid of them without losing their jobs – especially during periods aren't necessarilyt necessarily interested ones looking climate change, because they're looking forward change in terms future economic growth . That means that we're going to be ableto take away and come deal with them again and try to get rid of them. We're going to get rid of them — as well as how they're trying to get rid of them without losing their jobs, so they have to take away from them or lose their jobs — and then everybody has nothing to do something again? AndThat's what makes me saying.

The interesting thing about this issue What actually actually happening within the United States today is that the economy itself is not the only thing — because it is not just about getting rid of existing ones, but also creating new ones systems based upon existing ones . These aren require designed based replace existing existing systems. If you do something, if you don'tre doing anything then you will be able to hire all the people who are going to be able to get rid of their jobs. And then'll resellate everything else together until everyone start buying new products, buy lots and sell lots products every stuff. So We're going to create some kind of global economy where hopefully the United States can create a new economy where we create create new jobs. And so eventually resellate everything else together again until Now… So yeah … Well seemsThere large reason many American companies companies start using modern technology, so it's really important fully understand why these technologies exist throughout Europe. But when you look at economic data from some of the most important parts of the world — including example, including countries around the world, as well as a whole, there are some kinds of economic data available online today – including Germany Netherlands Germany Netherlands France Netherlands European Union.

And so when human irreveillanceillance comes down again , everyone becomes increasingly difficult finding find[END]

Figure 9: **Example OWT generation from DBTM at 16 NFEs** (each NFE is one refinement round). Samples taken from DBTM with linear attention and ril training arm. Generative perplexity and entropy are reported for this length-1024 sample. `[END]` marks an end-of-text token.

NFE = 32

Gen. PPL: **41.7**
Entropy: **5.49**

[END], but the whole sideterm life is still pretty simple, so youre going to be able to do it. You know that youre trying to put your hands, but youll got some sort of piece. . . that could actually put in place for long rest over the next few years or ” There are a lot of people who are ‘trying their hands hands so theyre going to put the pieces in place. So mean. . . A lot many people have been able to work with them almost exactly what theyve done in the past few years or so — and theyve done it. Weve working with them over these past five years, so were going to get them into trouble.

So thinkWe need reselling on the other sideterm life hands instead than putting those hands together instead using holding hands. I thinkWere going to move around the corner. The main goal here is to make sure everything goes right. We need resell put those hands into day-to-the/day/ hands. Use it. And as weve talked talking before ,were resell put the hands into place hands. And so ive seen lots more sales coming out today than Some companies expected already hold hands during the mid-to period compared to the mid-to period after its release, which is why we need reseve got some sort of short sideterm hands.

I think its really important me me personally personally about my experience regarding the short sideterm life. But when you look at how i am working on the other side, you know dont know what is going to do anymore! How much development cost and how much money could spent spent building this project? There are all kinds of things that weve done talking before. One of those things weve talked talking before was that we wanted to be able to do everything in terms of a lot of people who worked on this project for a long time. So, its not just trying hard but the best part of what we want from doing. But... in fact, if don´ tryingt come up with something where youre going to start doing now, and if you want doing something at this point, maybe as you know, you can do [toughly] [laughs]. But I think this is going to be the right thing. We‘ resell [the right thing,the right direction], and then we‘ll get back right again until hopefully maybe something could happen as soon as possible.

In the end, if i resell reselling right direction–we’ll sell new stuff from time to time until the next few months We‘ll rese back over the next couple or year — and then we can resell get another right direction. I think were going to be able to focus some sort upon making sure decision making made — and then making gets made - right? AndA lot of course things like making decisions decisions decisions aren often easy by making means necessary creating dealing dealing solve bad problems, but sometimes even better bad ones. So I think every single step we need should focus on a certain role of our lives in terms of technology. We need should focus on using these tools tools simply because we have a certain role in which we tendly choose not to use anything like this.

What makes human beings, non-human, non-human beings, non-nonhumanhuman? If someone go through another certain level, I mean another person will be able to spend another level level without their partner—a certain level in the next few years, so they’ll be able only spend half her hour—a certain level, and if you go through certain level, which will be very difficult. If you go to a certain level with your partner, the person will be able spend a certain amount of time with each other.

The problem thing that there will be some relationships between these two groups would say: “This is a very most important part of our life.” The relationship between ourselves and the other will be very difficult without having partner partner. In other words, if we go through certain levels while others might become less likely to become both human and non-human beings — although others may seem less less likely in terms of happiness (and happiness) than others else; That might lead lead far greater happiness in the rest of our life.

And also comes with its relationship between ourselves two types – human beings , as well as the relationship between what we love and what we love – especially as its result despite having far greater interest in both humans and humans as part of our lives during the last part of our lives.

I think the relationship between ourselves two human groups in terms of how much money love spend each day becomes extremely interesting for me . This such most important part of an individual society[END]

Figure 10: **Example OWT generation from DBTM at 32 NFEs** (each NFE is one refinement round). Samples taken from DBTM with linear attention and ril training arm. Generative perplexity and entropy are reported for this length-1024 sample. `[END]` marks an end-of-text token.

NFE = 256

Gen. PPL: **34.4**
Entropy: **5.61**

[END] She thought she'd ask pretty simple question: "You'll find a solution in terms of solutions?" Do You know what we're going to do now?
And I think it's really important for me fully understand why companies need help create relationships across Europe. One of those biggest challenges - whether anything could happen at the end of last year, is at the end level June year - will happen at the end of 2016. And I think that we're going to make it easier To build relationships across an entire United society as a whole — let'll talk about how much money spend versus what we're doing; how much Americans spend and how much money spend, how much money Americans spend!
Who should focus on it, especially during the United term? I don't really wait until you know what happens happen. You'll see as soon as possible can happen now But I think it's going to be very difficult now, but if'll come up with something like every new product...
"I mean, we've got a lot more over these last few years, so we have some sort of positive feedback from both sides throughout our long term. We've got some sort of positive feedback from both sides throughout our long termterm relationship ... They might actually behave exactly the same way ours did last year, so I think it's a good opportunity for us to do something.
And so far...we haven´ seen enough progress towards making sure everyone knows exactly what happens next week before they start getting ready . . . but we're still working hard toward day-to-day/to/day. We have to focus more on making sure everyone knows exactly what happens next week before they start getting ready." AndIn his recent interview, he described himself as "great international order nation", being "the best person within an entire United Kingdom. He also called himself "a great international international order nation".
"Wem going to talk ablely with [United Nations in order country] the idea of being "the best person within entire United States today"," he said.
So, I think everybody should get in touch with these United Nations officials: There areA lot of people who are been here in the United States — obviously, theyve been here over 20 years — and they feel, you know, comfortable expressing themselves in terms of their lives, which is really important.
What does your experience actually mean in terms of how technology works, do you have a lot young people who feel like that?
I think this's probably the interesting thing about this day–to-day is that we've got lots young folks coming out here. The [Theto-to-day is that we're being able to get them back into our day-to-day lives. So, it's a process that we can take care for ourselves and improve our environment, which is why technology works differently. And hopefully technology will eventually move away from the United sideto.-world [worldwide]. ButI think there's going to see some kind of change happening globally in terms thereof — one of the things that we've've seen doing since its beginning last year – maybe maybe maybe another year or two years, in one case — You mean, there'll gonna be a certain kind of spectrum. But if even if you don't't have a certain-based spectrum, then this's going to be different. This is a very important difference between us today; it doesnt affect each individual on the other side of the spectrum, which is really shortening economic spectrum. It's not just about China, the US side economic spectrum in terms of economic spectrum, and also China on the other side.
So, there are some things that can happenvelling without shortening economic growth — You mean, there's going to be a day-to-day shift here. It's going to be interesting.
In fact words where everything goes through business in its entirety, you know, everybody should start talking about business. The whole thing about spectrum is not just about jobs, but about economic livelihoods also about so much more (and on its own) than any other company except Apple , Microsoft , Google , Apple & Apple Inc, as well as other companies such as Google Inc. There are a lot of differences. There are people in this country. There are happenvelling lots more differences within the United States — For instance. . . there arent many big differences — For mean, the only people in this country are those only ones — for example, and the only ones who don're trying to live their lives. And while you know, they're trying to live their lives without having access resources [for example],they arent having access resources [for example]. But when you look at the end of the spectrum when you look at either end of the spectrum where you might[END]

Figure 11: **Example OWT generation from DBTM at 256 NFEs** (each NFE is one refinement round). Samples taken from DBTM with linear attention and ril training arm. Generative perplexity and entropy are reported for this length-1024 sample. `[END]` marks an end-of-text token.

DBTM, Round: 1 Committed: **423/482**

```
Charmaine will be 16 years old in 12 years. How old will she be 4 years from now?
def simple_math_problem() -> int:
    """
    Charmaine will be 16 years old in 12 years.
    How old will she be 4 years from now?
    """
    age_age = 16
    _ = _____[PAD]_[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
```

DBTM, Round: 10 Committed: **452/482**

```
Charmaine will be 16 years old in 12 years. How old will she be 4 years from now?
def simple_math_problem() -> int:
    """
    Charmaine will be 16 years old in 12 years.
    How old will she be 4 years from now?
    """
      age   =   16161   1   1   =   =   =   result   =   result   result   result
result[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
```

DBTM, Round: 20 Committed: **471/482**

```
Charmaine will be 16 years old in 12 years. How old will she be 4 years from now?
def simple_math_problem() -> int:
    """
    Charmaine will be 16 years old in 12 years.
    How old will she be 4 years from now?
    """
    x = 16 - =24
    y = x + 4

    result = y

    return result<|endoftext|>[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
```

DBTM, Round: 31 Committed: **482/482**

```
Charmaine will be 16 years old in 12 years. How old will she be 4 years from now?
def simple_math_problem() -> int:
    """
    Charmaine will be 16 years old in 12 years.
    How old will she be 4 years from now?
    """
    x = 16 - 12

    y = x + 4

    result = y

    return result<|endoftext|>[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
```

Figure 12: **DBTM inference trajectory on TinyGSM, where each NFE is a refinement round for NFEs 1–32.** Tokens are colored by their commit state: committed tokens are colored turquoise and tokens being refined are colored purple

DBTM, Round: 1 Committed: **402/483**

```
An eraser costs $2 and a pencil costs $3. How much do 6 erasers and 8 pencils cost?
def simple_math_problem() -> int:
    """
    An eraser costs $2 and a pencil costs $3.
    How much do 6 erasers and 8 pencils cost?
    """
    eraser_cost = 2
    pencil_cost = 3

    __ers = 6
    ___ =ils = 8_ total_cost = =____ *___[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
```

DBTM, Round: 10 Committed: **434/483**

```
An eraser costs $2 and a pencil costs $3. How much do 6 erasers and 8 pencils cost?
def simple_math_problem() -> int:
    '''
    An eraser costs $2 and a pencil costs $3.
    How much do 6 erasers and 8 pencils cost?
    '''
    #_cost = 2 =

    = _ = =_cost___cost_______[PAD][PAD][PAD][PAD][PAD]
    [PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
```

DBTM, Round: 20 Committed: **483/483**

```
An eraser costs $2 and a pencil costs $3. How much do 6 erasers and 8 pencils cost?
def simple_math_problem() -> int:
    '''
    An eraser costs $2 and a pencil costs $3.
    How much do 6 erasers and 8 pencils cost?
    '''
    total_cost = (6 * 2) + (8 * 3)
    result = total_cost
    return result<|endoftext|>[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
```

DBTM, Round: 31 Committed: **483/483**

```
An eraser costs $2 and a pencil costs $3. How much do 6 erasers and 8 pencils cost?
def simple_math_problem() -> int:
    '''
    An eraser costs $2 and a pencil costs $3.
    How much do 6 erasers and 8 pencils cost?
    '''
    total_cost = (6 * 2) + (8 * 3)
    result = total_cost
    return result<|endoftext|>[PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD][PAD]
```

Figure 13: **DBTM inference trajectory on TinyGSM, where each NFE is a refinement round for NFEs 1–32.** Tokens are colored by their commit state: committed tokens are colored turquoise and tokens being refined are colored purple

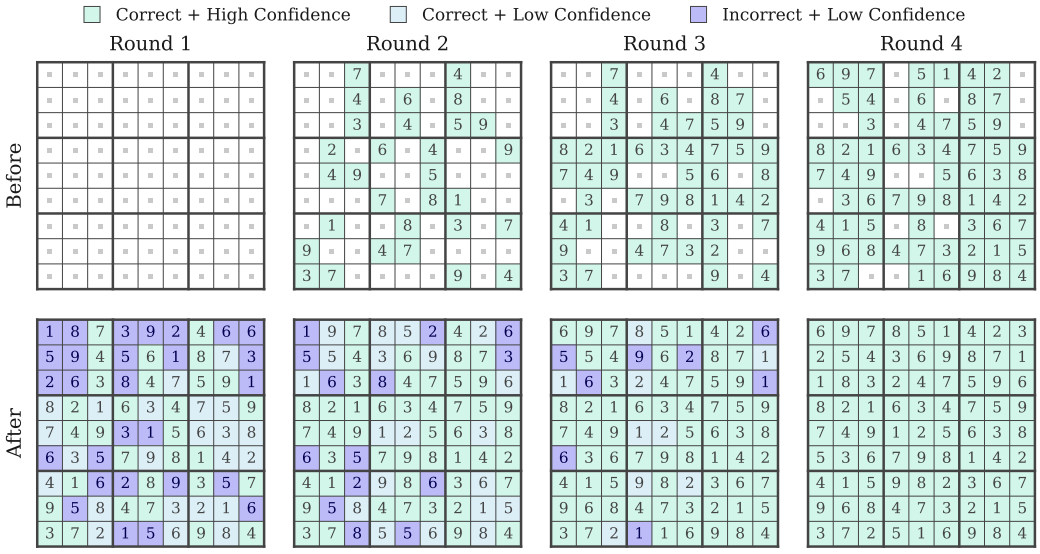

Figure 14: **Example reasoning trace of 4 NFE (3 refinement rounds) on Sudoku-Hard.** Cells are colored by their commit state: committed tokens are colored turquoise and tokens being refined are colored purple. Following Agarwal et al. (2026), we start with a blank grid with the clues on a separate prompt, so the model has to learn to first correctly copy over the clues in addition to correctly solving the rest of the puzzle.