CONSTRAINED TABULAR GENERATION: EXACT CONDITIONING OF ARBITRARY MARKOV PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular data generation supports privacy-preserving data sharing, minority-class augmentation, test-database generation, and missing-value imputation. These often require generated rows to satisfy constraints on arbitrary subsets of columns with probability one, while matching the distribution of real rows satisfying them. We study this as *constrained tabular generation*. Existing methods are limited to pinned values, sacrifice distributional fidelity for constraint satisfaction, or restrict exact conditioning to continuous diffusions. Using generator matching, we unify tabular flow, diffusion, and jump models through Markov generators and introduce TABJUMP, a pure-jump model for both numerical and categorical columns. We apply Doob's h-transform to general Markov generators, deriving exact conditioning through additive drift corrections for diffusions and multiplicative reweighting of jump rates, both learnable with a consistent objective. This characterizes which processes admit corrections: deterministic flows do not, since the transform leaves the generator unchanged, although in special cases flows admit training-free but biased conditioning using a mean-field approximation of their endpoint posterior. We amortize conditioning through a trainable guide that takes constraints as input, keeps the unconditional generator frozen, and generalizes to unseen constraints across arities, modalities, types, and selectivities. Across eight datasets, our methods improve unconditional fidelity and outperform imputation-, penalty-, and LLM-based baselines in both constraint satisfaction and conditional fidelity.

1 INTRODUCTION

Tabular data generation enables privacy-preserving sharing (Cătălina Stoian et al., 2025; Kulynych et al., 2026; McKenna et al., 2022), minority-class augmentation (Sauber-Cole & Khoshgoftaar, 2022; Engelmann & Lessmann, 2021), workload-driven test-database generation (Binnig et al., 2007; Lo et al., 2010), and missing-value imputation (Zheng & Charoenphakdee, 2022; Zhang et al., 2025). These applications require more than the *unconditional* generation addressed by recent diffusion (Kotelnikov et al., 2023; Shi et al., 2025; Zhang et al., 2024) and flow-matching (Guzmán-Cordero et al., 2025; Nasution et al., 2025) models. Rather, they require generation under *constraints* on selected columns, such as minority labels, query predicates, or observed entries.

Consider the canonical table operation `SELECT * WHERE` S, which may combine range, set-membership, and equality constraints over arbitrary columns. Its generative counterpart should accept any such S and generate rows satisfying it while following the true conditional $p_{\mathrm{data}}(\cdot \mid S)$. Existing methods fall short: classifier and classifier-free guidance (Dhariwal & Nichol, 2021; Ho & Salimans, 2022) do not guarantee satisfaction, manifold guidance (Shankar et al., 2026) does not guarantee the true conditional, imputation (Zheng & Charoenphakdee, 2022; Zhang et al., 2025) naturally handles only pinned values, and rejection sampling is impractical for rare constraints.

Recent diffusion and flow-matching models for table generation can be viewed as Markov processes evolving from noise to data, with various combinations of diffusion, flow, and jump dynamics, designed to accommodate heterogeneous, strongly dependent columns. Doob's h-transform provides a principled route to conditioning Markov processes exactly on a terminal event S, but existing learning-based approaches largely focus on continuous diffusions (Denker et al., 2024; Guo et al., 2026). Applying this principle to tables requires a unified treatment of mixed dynamics at the generator level, together with architectures that respect row- and column-order symmetry.

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.

Tabular generators as generator-matching instances (§3)	
Markov Generators on Mixed State Spaces	Theorem 3.1, App. C
A Generator View of Existing Models	Table 1
TABJUMP- A Pure Jump Model	§ 3.1, App. G.4
Order Dependence of Flattened cores; A new Symmetric core	Prop. G.4, App. G.3.2
Exact conditioning of Markov processes by Doob's h-transform (§4)	
Doob Conditioned Process, Generator, Parameters	Theorem 4.1, 4.2, Cor. 4.3
Linearity and Column-wise conditioning	Cor. 4.4, App. E.3
Markovian Projections	App. D
Learning with per-sample paths; Consistency	Prop. 4.5, Lem. 4.6, App. E.6, D
Flows: Invariance and a closed-form estimator	Cor. 4.7, App. F.1
Amortized conditioning and experiments (§5–6)	
Generator Corrections and New Losses	Eqs. 13, 14, App. H.2
Guide Modules, Training, Sampling, Cost	14, App. H
Datasets, constraint suites; evaluation	App. I, Table 2, Figure 2

Figure 1: Overview of the main results.

Contributions. We propose a general framework for exact conditioning of arbitrary Markov processes, building upon the principles of generator matching (Holderrieth et al., 2025), and apply it to constrained tabular generation. Our main theoretical (Figure 1) and empirical contributions are:

(1) Generator-level unification (§3). We unify existing tabular models through a common Markov-generator decomposition, and address two limitations. First, no current model uses jumps for numerical columns, despite their frequent discontinuities and point masses. In response, we propose TABJUMP, with a common jump mechanism for both numerical and categorical columns. Second, we identify a common source of column-order dependence in existing architectures and replace it with a permutation-equivariant core, which enforces column symmetry by construction.
(2) Exact conditioning of Markov processes (§4). We leverage the machinery of Doob's h-transform for general Markov generators to perform exact conditioning which allows us to modify the parameters of the generator. We then propose a scalable, model-agnostic learning framework with a consistent objective. Further, these insights lead to a natural characterization of processes which are amenable to conditioning.
(3) Amortized conditioning and evaluation (§5, §6). We amortize the cost of training over multiple constraint sets. Tables permit systematic variation of constraint selectivity and direct evaluation against real conditional subpopulations. On constraint suites spanning selectivities, column types, and arities, our methods outperforms baselines in both constraint satisfaction and conditional fidelity.

2 PRELIMINARIES AND NOTATION

Notation. A *numerical* column takes values in $\mathbb{R}$ and a *categorical* column in a finite set of categories. Let E_j denote the state space of column j and $E := \prod_j E_j$ the state space of a row. Grouping columns by type, $E = \mathbb{R}^m \times F$, where m is the number of numerical columns and $F = \prod_{j \text{ categorical}} E_j$ is the finite set of joint categorical assignments. We write $x = (x^{\text{num}}, x^{\text{cat}}) \in E$ for a row split by type, p_{data} for the data distribution on E, $S \subseteq E$ is the set of rows satisfying a collection of column-wise constraints, with $p^S_{\text{data}} := p_{\text{data}}(\cdot \mid S)$ the conditional distribution.

Markov processes and generators. We consider a Markov process $X = (X_t)_{t\in[0,1]}$ on E with path $\mathbb{P}$, transition function $p(s,t,x,\mathrm{d}y) = \mathbb{P}(X_t \in \mathrm{d}y \mid X_s = x)$ for $s \le t$, and marginal distribution p_t. Time runs from noise at $t = 0$ to data at $t = 1$, such that $p_1 = p_{\text{data}}$. The infinitesimal evolution of a markov process is described by a linear operator, called the *generator* $\mathcal{L}_t$, through its action on a domain $\mathcal{D}(\mathcal{L}_t)$ of test functions f such that

$$\mathcal{L}_t f(x) \;:=\; \lim_{\Delta \downarrow 0} \frac{\mathbb{E}[f(X_{t+\Delta}) \mid X_t = x] - f(x)}{\Delta}, \tag{1}$$

If the marginals satisfy the Kolmogorov forward equation $\partial_t \int_E f \, \mathrm{d}p_t = \int_E \mathcal{L}_t f \, \mathrm{d}p_t$ for all $f \in \mathcal{D}(\mathcal{L}_t)$ (Appendix B.4); we say that X *generates* the path $(p_t)_{t\in[0,1]}$. Further, knowledge of $\mathcal{L}_t$ suffices to simulate X (Holderrieth et al., 2025, Algorithm 2). More details are presented in Appendix B.

Generator matching (Holderrieth et al., 2025) constructs the process X from per-sample processes. For each $z \sim p_{\mathrm{data}}$, they fix a *per-sample path* $p_t(\cdot \mid z)$ from a reference distribution p_0 to δ_z (the point mass at z), whose generator $\mathcal{L}_t^z$ is analytically known. Writing $p_{1|t}(\mathrm{d}z \mid x) \propto p_t(x \mid z)\, p_{\mathrm{data}}(\mathrm{d}z)$ for the endpoint posterior, the marginal path $p_t = \int_E p_t(\cdot \mid z)\, p_{\mathrm{data}}(\mathrm{d}z)$ is then generated by

$$\mathcal{L}_t f(x) \;=\; \mathbb{E}_{z\sim p_{1|t}(\cdot|x)}\big[\mathcal{L}_t^z f(x)\big] \tag{2}$$

Importantly, this construction matches marginal paths, but not necessarily the joint distributions of intermediate states and endpoints. In particular, $p_{1|t}(\cdot \mid x)$ need not equal $p(t, 1, x, \cdot)$. They can differ for deterministic flows, whose endpoint is fixed by x, whereas $p_{1|t}(\cdot \mid x)$ may still assign mass to multiple endpoints.

Rather than learning the operator directly, we parameterize $\mathcal{L}_t$ by components $F_t(x)$ independent of the test function f (Theorem 3.1), By linearity, Eq. 2 gives $F_t(x) = \mathbb{E}_{z\sim p_{1|t}(\cdot|x)}[F_t^z(x)]$. Conditional generator matching (CGM) (Holderrieth et al., 2025, Prop. 2) learns F_t by regressing a network F_t^θ against the per-sample parameters F_t^z. Here, "conditional" refers to the endpoint z, rather than our constraint set S. The objective minimizes a loss $L_{\mathrm{cgm}}(\theta)$ which is given as

$$L_{\mathrm{cgm}}(\theta) \;=\; \mathbb{E}_{t\sim\mathrm{Unif}[0,1],\; z\sim p_{\mathrm{data}},\; x\sim p_t(\cdot|z)}\big[D\big(F_t^z(x), F_t^\theta(x)\big)\big], \tag{3}$$

where D is a Bregman divergence. This objective has the same gradients as the intractable regression against F_t, and therefore the same minimizer (Holderrieth et al., 2025, Prop. 2).

3 Tabular Generative Models as Generator-Matching Instances

We characterize Markov generators on mixed state spaces, which unify existing tabular models and motivate TabJump (§3.1). We then replace the order-sensitive network core shared by existing models with a column-symmetric one (§3.2).

Theorem 3.1 (Universal characterization of generators on mixed state spaces). *Under the regularity assumptions of Appendix B.4, the generator of a Markov process on $E = \mathbb{R}^m \times F$ satisfies*

$$\mathcal{L}_t f(x) \;=\; \nabla_{\mathrm{num}} f(x)^\top u_t(x) \;+\; \tfrac{1}{2}\nabla^2_{\mathrm{num}} f(x) \cdot \sigma_t^2(x) \;+\; \int_E \big[f(y) - f(x)\big]\, Q_t(\mathrm{d}y, x), \tag{4}$$

with velocity $u_t(x) \in \mathbb{R}^m$, positive semi-definite diffusion matrix $\sigma_t^2(x)$ (with $\cdot$ the Frobenius inner product), and jump kernel $Q_t(\cdot, x)$, a finite measure on E. Here ∇_{num} is with respect to x^{num}. Flow and diffusion act only on numerical columns, while jumps may happen at both types of columns. We call the triplet (u_t, σ_t^2, Q_t) the parameters *of X.*

Sketch: We view E as $|F|$ slices of $\mathbb{R}^m$ and show that transitions between slices can only be jumps. As special cases, when $m = 0$, Q_t is the rate matrix of a continuous-time Markov chain (CTMC) and when $|F| = 1$, Eq. 4 recovers the Euclidean case. It strictly generalizes Holderrieth et al. (2025, Thm. 1 and Prop. 5) by allowing cross-domain simultaneous jumps.

Tabular models. The models considered here use per-sample paths that factorize across columns, yielding $\mathcal{L}_t = \sum_j \mathcal{L}_t^{(j)}$, where each $\mathcal{L}_t^{(j)}$ is the component-wise generator acting on one column with parameters that depend on the full row x (Holderrieth et al., 2025, Props. 4 and 5). Consequently, σ_t^2 is diagonal and each jump changes only one column, while the network captures cross-column dependencies through the generator parameters. Table 1 summarizes the components and CGM losses of the data-space models. Appendix G provides their details and extends the characterization to latent-space models.

3.1 TabJump: A pure-jump tabular generator

Theorem 3.1 suggests a modeling choice unexplored by existing tabular models, namely jumps on numerical columns. Numerical columns often exhibit discontinuities and large point masses, which

		TABDIFF	TABBYFLOW	TABJUMP
	State space E	$\mathbb{R}^m \times \bar{F}$	$\mathbb{R}^{m+\sum_j \lvert E_j \rvert}$	$\prod_{j\,\text{numerical}} (\mathbb{R} \sqcup \{\mathsf{M}\}) \times \bar{F}$
Numerical cols.	Components (Eq. 4)	flow + diffusion (reverse SDE) $u_t(x) = g_t^2 \nabla \log p_t(x), \quad \sigma_t^2 = g_t^2$	flow $u_t(x) \in \mathbb{R}^m, \quad \sigma_t^2 = 0$	jump (masked, mixture kernel) target $Q_t(\mathrm{d}y, x)_j$ nonzero only if $x_j = \mathsf{M}$
	Generator	$\nabla f(x)^\top u_t(x) \; + \; \frac{1}{2} \nabla^2 f(x) \cdot \sigma_t^2$	$\nabla f(x)^\top u_t(x)$	$\int_{\mathbb{R}} \big(f(y) - f(x)\big)\, Q_t(\mathrm{d}y, x)_j$
	CGM loss	$\big\lVert s_\theta(x_t, t) - \nabla \log p_t(x_t \mid z) \big\rVert^2$	$\big\lVert u_\theta(x_t, t) - \frac{z - x_t}{1-t} \big\rVert^2$	generalized KL of the kernels, equal to the categorical entry on bins
Categorical cols.	Components (Eq. 4)	jump (masked CTMC) $Q_t(y, x)$, nonzero only if $x_j = \mathsf{M}$	flow (one-hot vectors) $u_t(x) \in \mathbb{R}^{\lvert E_j \rvert}, \quad \sigma_t^2 = 0$	jump (masked CTMC) $Q_t(y, x)$, nonzero only if $x_j = \mathsf{M}$
	Generator	$\sum_y \big(f(y) - f(x)\big)\, Q_t(y, x)$	$\nabla f(x)^\top u_t(x)$	$\sum_y \big(f(y) - f(x)\big)\, Q_t(y, x)$
	CGM loss	$\sum_{y \neq x} \big(Q_t^\theta(y, x)$ $-Q_t(y, x \mid z) \log Q_t^\theta(y, x)\big)$	$-\log \mu_\theta(x_t, t)_z$	$\sum_{y \neq x} \big(Q_t^\theta(y, x)$ $-Q_t(y, x \mid z) \log Q_t^\theta(y, x)\big)$

Table 1: **Tabular generation models as instances of Thm. 3.1.** Per column type, we show the active components of Eq. 4, the parameters, generator on a test function f, and CGM loss (Eq. 3). $\bar{F}$ adds a mask state M to every categorical column; g_t is the diffusion coefficient, s_θ, u_θ, μ_θ, Q_t^θ the learned score, velocity, endpoint distribution, and jump kernel, and $Q_t(\cdot \mid z)$ the per-sample kernel. In jump terms, y ranges over column j and $f(y)$ sets column j to y. Table 4 gives the full version.

continuous dynamics do not naturally support. Jump kernels are flexible, with Gaussian (Bertazzi et al., 2024) and α-stable Levy (Yoon et al., 2023) kernels explored in other modalities. We exploit this to introduce TABJUMP, a pure-jump generator for numerical and categorical columns alike.

We quantize each numerical column into at most B quantile bins, retaining frequent repeated values as singleton bins. Following Holderrieth et al. (2025), interval bins have uniform distributions, while singleton bins correspond to Dirac measures, yielding jump kernels composed of both. Formally, each numerical column lives in $\mathbb{R} \times \{0, 1\}$, with transitions $(x, 1) \to (y, 0)$ only. When $Q_t(\mathrm{d}y, 0; x, 1)$ is independent of x, $\mathbb{R} \times \{1\}$ acts as a mask state M. The row can then be regarded as a masked CTMC over numerical bin sets and categorical alphabets, each augmented with M, trained using TABDIFF's categorical loss. Both column types share the same mechanism and loss, while numerical intervals and categorical constraints reduce to subsets of finite alphabets and can be conditioned in the same manner (Section 4). Singleton bins preserve repeated values exactly, while uniform decoding of interval bins limits variation below the bin resolution. Further details are presented in Appendices G.4 and G.5.

3.2 A COLUMN-SYMMETRIC CORE FOR TABULAR GENERATORS

The models in Table 1 share a common architecture. Since columns evolve separately given the joint state, their network core captures cross-column dependencies. A noisy row $x = \{x_j\}_{j=1}^n$ at time t is mapped to per-column parameters $\Phi_\theta(x, t) = \{\phi_j\}_{j=1}^n$ through $r_j = \textsc{Tokenizer}_{\theta_j}(x_j)$, $R' = \textsc{Core}_{\theta_0}(R, t)$, and $\phi_j = \textsc{Detokenizer}_{\theta_j}(r'_j)$, where $R = [r_1; \ldots; r_n] \in \mathbb{R}^{n \times d}$, θ_j are per-column parameters, and θ_0 is shared (Appendix G.3.1). Since column order carries no information, permuting the inputs should only permute the outputs. However, the TABDIFF core is

$$\textsc{Core}_{\theta_0}(R, t) = \textsc{Dec}_{\theta_0}\Big(\mathrm{vec}^{-1}\big(\mathrm{MLP}_{\theta_0}(\mathrm{vec}(\bar{R}), t)\big)\Big), \qquad \bar{R} = \textsc{Enc}_{\theta_0}(R), \tag{5}$$

where ENC and DEC are self-attention blocks without positional encodings, and vec flattens tokens in column order. TABBYFLOW similarly applies an MLP to $\mathrm{vec}(x)$. These MLPs tie weights to column positions, breaking permutation equivariance (Appendix G.3.2).

The symmetric core. We replace the flattened MLP with stacked self-attention blocks that directly model cross-column interactions. For block ℓ, starting from $R^{(0)} = R$,

$$\tilde{R} = R^{(\ell)} + \textsc{Attn}_{\theta_0}\big(\mathrm{LN}(R^{(\ell)} + \mathbf{1}_n e_t^\top)\big), \qquad R^{(\ell+1)} = \tilde{R} + \mathrm{FFN}_{\theta_0}\big(\mathrm{LN}(\tilde{R})\big), \tag{6}$$

where e_t is a sinusoidal time embedding (Ho et al., 2020), ATTN is multi-head self-attention without positional encodings, LN is layer normalization, and FFN acts per column. Every operation commutes with column permutations, making the core equivariant by construction (Proposition G.4). It replaces the cores of TABDIFF and TABBYFLOW, while also serving as the core of TABJUMP (§6).

4 EXACT CONDITIONING OF MARKOV PROCESSES

We condition any Markov process of Theorem 3.1 on its endpoint. We use the Doob's h-transform to map parameters of an unconditional diffusion, jump process, or mixture to their exact conditional counterparts (§4.1). We show that they are learnable by generator matching using only rows satisfying S and the original per-sample paths (§4.2). Deterministic flows admit no such correction, but in special cases an approximate endpoint posterior yields a training-free, albeit biased conditioner (§4.3).

4.1 EXACT CONDITIONING OF A MARKOV PROCESS ON ITS ENDPOINT

We seek a process X^S whose endpoint follows p^S_{data}, so that $X^S_1 \in S$ almost surely. Assume $\mathbb{P}(X_1 \in S) > 0$ and let $h(t,x) \coloneqq p(t,1,x,S) = \mathbb{P}(X_1 \in S \mid X_t = x)$. By the Markov property, $h(t, X_t)$ is a (local) martingale, and reweighting the distribution of X on $[0,t]$ by $h(t,X_t)/\mathbb{P}(X_1 \in S)$ yields a process assigning zero measure to trajectories ending outside S. This change of measure is called the **Doob's h-transform** (Doob, 1957). A key property of this technique is that it provably allows sampling from the true conditional p^S_{data}. We extend the proof of Guo et al. (2026) to Markov processes with possibly discontinuous paths and show the following theorem:

Theorem 4.1 (Endpoint-conditioned process). *Let X satisfy the regularity assumptions of Appendix B.4 and let $S \subseteq E$ be measurable with $\mathbb{P}(X_1 \in S) > 0$. The reweighted process, denoted by X^S, satisfies: (i) X^S is Markov; (ii) its transition function and initial distribution are*

$$p^S(s,t,x,\mathrm{d}y) = p(s,t,x,\mathrm{d}y)\,\frac{h(t,y)}{h(s,x)}, \qquad p^S_0(\mathrm{d}x) = p_0(\mathrm{d}x)\,\frac{h(0,x)}{\mathbb{P}(X_1 \in S)}; \tag{7}$$

(iii) its terminal distribution is $p^S_{\text{data}} = p_{\text{data}}(\cdot \mid S)$, the true conditional distribution given S.

The proof is in Appendix E.1. We highlight the following properties of Eq. 7: **(1)** The factor $h(t,y)/h(s,x)$ restricts X^S to $\{h > 0\}$, where the process remains Feller; **(2)** The initial distribution is reweighted by $h(0,\cdot)$. For diffusions and masked chains started from pure noise, X_1 is nearly independent of X_0, so $h(0,\cdot)$ is nearly constant and $p^S_0 \approx p_0$. For a flow, X_1 is determined by X_0, so the dynamics are unchanged and conditioning reduces to reweighting p_0 (§4.3).

Theorem 4.2 (Generator of the conditioned process). *Under the regularity assumptions of Appendix B.4, for every test function $f \in \mathcal{T}$ and every x with $h(t,x) > 0$, the generator of X^S is*

$$\mathcal{L}^S_t f(x) \;=\; \frac{\mathcal{L}_t\big(h(t,\cdot)f\big)(x) - f(x)\,\mathcal{L}_t h(t,\cdot)(x)}{h(t,x)}. \tag{8}$$

(Corstanje et al., 2023, Thm 3.3) establishes a similar result, but with a key distinction. We additionally prove that the conditioned process remains Feller, allowing us to apply Theorem 3.1. Further details are present in Appendix A.2. As mentioned above, X^S satisfies the regularity assumptions of Appendix B.4, and so Theorem 3.1 applies to it, and substituting Eq. 4 into Eq. 8 yields the conditioned parameters.

Corollary 4.3 (Universal characterization of conditioned parameters). *Let X have parameters (u_t, σ_t^2, Q_t) in (4), and let $h(t,\cdot)$ be twice continuously differentiable in x^{num} on $\{h > 0\}$. Then the parameters of X^S are obtained from those of X as*

$$u^S_t = u_t + \sigma_t^2\,\nabla_{\text{num}} \log h(t,\cdot), \qquad \sigma^{2,S}_t = \sigma^2_t, \qquad Q^S_t(\mathrm{d}y,x) = Q_t(\mathrm{d}y,x)\,\frac{h(t,y)}{h(t,x)}. \tag{9}$$

We highlight three consequences. First, conditioning modifies only the velocity and jump rates, preserving the diffusion coefficient and jump support. The original sampler with corrected parameters thus yields samples in S almost surely (Theorem 4.1). Second, these corrections recover Girsanov's drift change, approximated by classifier guidance (Dhariwal & Nichol, 2021) for Euclidean spaces, and the corresponding rate reweighting for jump processes (Corstanje et al., 2023). Third, Eq. 9 and Theorem 3.1 also cover jump diffusions (Campbell et al., 2023) and piecewise deterministic processes (Bertazzi et al., 2024). For pure flows, both corrections vanish (§4.3).

For tables, the generators of §3 decompose into per-column terms. By linearity of Eq. 8 in $\mathcal{L}_t$, conditioning preserves this decomposition without introducing joint jumps or correlated noise.

Corollary 4.4 (Multimodal splitting of the conditioned generator). *Given $E = \prod_j E_j$ and $\mathcal{L}_t = \sum_j \mathcal{L}_t^{(j)}$, where $\mathcal{L}_t^{(j)}$ updates only column j, then $\mathcal{L}_t^S = \sum_j \mathcal{L}_t^{S(j)}$, with each $\mathcal{L}_t^{S(j)}$ obtained from $\mathcal{L}_t^{(j)}$ by* (8)*. For tables, the per-column parameters follow from* (9) *applied to column j.*

The conditioned process is simulated column by column as before, with each update depending on the full row through the model and h. For instance, a jump of column j to value y has conditioned rate $Q_t(y, x)\, h(t, x')/h(t, x)$, where x' is the row x with column j set to y (Proof in Appendix E.4).

4.2 Learning the conditioned parameters by generator matching

Corollary 4.3 expresses the conditioned parameters through the unknown function h, which we show is learnable by generator matching. The per-sample paths are unchanged by conditioning on S, since S is determined by the endpoint z and hence $p_t(\cdot \mid z, S) = p_t(\cdot \mid z)$. Conditioning only changes the endpoint posterior to $p^S_{1|t}(\mathrm{d}z \mid x) \propto p_t(x \mid z)\, p_{\mathrm{data}}(\mathrm{d}z)\, \mathbf{1}[z \in S]$.

Proposition 4.5 (Conditioned generator matching). *The process started from p_0 with generator*

$$\widetilde{\mathcal{L}}_t^S f(x) = \mathbb{E}_{z \sim p^S_{1|t}(\cdot|x)} \big[\mathcal{L}_t^z f(x)\big] \tag{10}$$

has terminal distribution p^S_{data}. If the per-sample paths X^z are the paths of the Markov process X constrained to hit z and $p_{1|t}(\cdot \mid x) = p(t, 1, x, \cdot)$, then $\widetilde{\mathcal{L}}_t^S = \mathcal{L}_t^S$ and $p_0^S = p_0$, so this process is precisely X^S, the Doob-transformed process of Theorem 4.1.

The first claim follows from Eq. 2 applied to p^S_{data}. The second requires the equality $p_{1|t}(\cdot \mid x) = p(t, 1, x, \cdot)$ of §2, which holds for diffusions and masked chains and fails for deterministic flows (§4.3). Further, the existence of the process X_t is asserted as the time-reversal of the standard forward noising process. The proof is presented in Appendix E.5, and further discussions on Markovian Projections are presented in Appendix D. Now, let $\widetilde{F}_t^S$ and F_t^S parameterize $\widetilde{\mathcal{L}}_t^S$ and $\mathcal{L}_t^S$, and let $F_t^{S,\theta}$ denote a neural approximation of $\widetilde{F}_t^S$. Following (3), we minimize

$$L^S_{\mathrm{cgm}}(\theta) = \mathbb{E}_{t \sim \mathrm{Unif}[0,1],\ z \sim p^S_{\mathrm{data}},\ x \sim p_t(\cdot|z)} \big[D\big(F_t^z(x), F_t^{S,\theta}(x)\big)\big]. \tag{11}$$

Compared with (3), only the endpoint z is restricted to rows satisfying S; the per-sample paths and targets F_t^z are unchanged.

Lemma 4.6 (Consistency of conditioned generator matching). *The loss $L^S_{\mathrm{cgm}}(\theta)$ equals $\mathbb{E}_{t,\, z \sim p^S_{\mathrm{data}},\, x \sim p_t(\cdot|z)}[D(\widetilde{F}_t^S(x), F_t^{S,\theta}(x))]$ up to a constant independent of θ. Its minimizer is therefore $\widetilde{F}_t^S$, whose generated process has terminal distribution p^S_{data}. When the condition of Proposition 4.5 holds, $\widetilde{F}_t^S = F_t^S$.*

This is obtained by applying Holderrieth et al. (2025, Prop. 2) to p^S_{data} (Appendix E.7). We keep the trained unconditional model F_t^θ fixed and learn only the correction in Eq. 9.

4.3 Conditioning deterministic flows

Corollary 4.7 (Invariance of pure flows). *If $\sigma_t^2 = 0$ and $Q_t = 0$, then $\mathcal{L}_t^S = \mathcal{L}_t$ on $\{h > 0\}$. Thus, X^S has the same velocity field as X and differs only through its initial distribution p_0^S.*

For a deterministic flow, X_1 is determined by X_t, so $h(t, x) \in \{0, 1\}$. By Eq. 7, $p_0^S = p_0\, \mathbf{1}[h(0, \cdot) = 1]/p_{\mathrm{data}}(S)$, which is rejection sampling at $t = 0$ with acceptance rate $p_{\mathrm{data}}(S)$ and is impractical for rare constraints. Crucially, under the CondOT Path Lipman et al. (2022), $u_t(x \mid z)$ is a linear function of z. TabbyFlow leverages this by pushing the expectation over z inside the conditional giving a velocity defined through the endpoint posterior, $u_t(x) = (\mathbb{E}[z \mid X_t = x] - x)/(1 - t)$ (§3). By Proposition 4.5, taking the mean under $p^S_{1|t}$ yields a flow with terminal distribution p^S_{data}.

Since the velocity only depends through the mean, TabbyFlow approximates the endpoint posterior by a product of Gaussian and categorical distributions across columns. Since S also factorizes, the restricted mean $\mathbb{E}[z \mid X_t = x, S]$ is available in closed form through 1-D truncated-Gaussians

and renormalized categorical probabilities (Appendix F.1). This conditioner requires no training, sampling, or extra forward passes. It is, however a biased estimator, since the posterior need not be Gaussian. In the unconditional case, minimizing squared error indeed recovers the posterior mean, with the explicit posterior never being used anywhere. Eijkelboom et al. (2025) instead condition on a soft observation through a fixed-point iteration targeting a posterior mode. For hard constraints such as S, the conditional mean is available directly.

5 Amortized Exact Conditioning

Section 4 reduces exact conditioning to corrections determined by h. We amortize these corrections across constraint sets using a lightweight *guide* module with trainable parameters χ, while keeping the unconditional generator θ frozen (Appendix H). The module applies to any diffusion or jump generator with the columnwise structure of §3.

Architecture. We use a smaller version of the column-symmetric core of §3.2 (Table 6). Each constraint is encoded per column as s_j. Numerical intervals are represented by their centre and log-width, and categorical sets by the sum of their frozen category embeddings. An additional embedding indicates whether the column is constrained. The frozen tokenizer provides column tokens r_j, which are projected, combined with the time embedding e_t and constraint encoding s_j, and processed by the symmetric core:

$$\tilde{r}_j = \mathrm{SiLU}\big(W_1[Wr_j; e_t]\big), \qquad r'_j = \mathrm{SiLU}\big(W_2[\tilde{r}_j; s_j]\big), \qquad R'' = \textsc{Core}_\chi([r'_1; \ldots; r'_n], t). \tag{12}$$

The module is equivariant under joint permutations of columns and their constraints (Appendix H.1).

Generator corrections. By Corollaries 4.3 and 4.4, conditioning adds $\sigma_t^2 \nabla_{\mathrm{num}} \log h$ to the velocity and reweights the jump kernel by $h(t,y)/h(t,x)$. We therefore learn a numerical correction $\Delta_\chi(x,t,S) \in \mathbb{R}^m$ in noise units, $\Delta_\chi \approx -\varsigma_t \nabla_{\mathrm{num}} \log h$ with ς_t the diagonal matrix of noise levels, and a scalar $\log h_\chi(x,t,S)$, giving

$$\begin{aligned} u_t^{S,\theta,\chi}(x) &= u_t^\theta(x) - \sigma_t^2(x)\, \varsigma_t^{-1} \Delta_\chi(x,t,S), \\ Q_t^{S,\theta,\chi}(\mathrm{d}y, x) &= Q_t^\theta(\mathrm{d}y, x)\, \exp\big(\log h_\chi(y,t,S) - \log h_\chi(x,t,S)\big). \end{aligned} \tag{13}$$

The diffusion matrix and jump support remain unchanged. We predict the numerical correction separately because an accurate h need not yield an accurate $\nabla h/h$ (Guo et al., 2026). For columnwise jumps, y in Eq. 13 is the row $x^{j \leftarrow y}$ with column j set to y. For finite alphabets, a head $\ell_\chi(x,t,S)_j[y]$ predicts $\log h$ at all candidate rows $x^{j \leftarrow y}$ in one forward pass. For masked chains, conditioning reweights the endpoint probabilities while preserving the unmasking schedule. Sampling uses the original generator with corrected parameters (Appendices H.2 and H.4).

Training. We minimize conditional generator matching across constraint sets, using

$$L(\chi) = \mathbb{E}_{S,\, t,\, z \sim p_{\mathrm{data}}^S,\, x_t \sim p_t(\cdot|z)} \Big[\sum_j D_j\big(F_t^z(x_t)_j,\, F_t^{S,\theta,\chi}(x_t)_j\big) \Big], \tag{14}$$

where $F_t^{S,\theta,\chi}$ are the corrected parameters, D_j is the Bregman divergence of the base generator's per-column loss (Table 1), and only χ is trained. By Proposition 4.5, we reuse the unconditional per-sample paths and targets F_t^z, restricting training to rows satisfying S. Each term is thus the original per-column loss evaluated on corrected parameters. For noise-prediction parameterizations, this generalizes the objectives of Denker et al. (2024) to other Bregman divergences (Appendix H.2). By Lemma 4.6, the population minimizers recover the exact corrections of Eq. 9 when the frozen generator is exact. Constraint sampling and training details are in Appendices I.3 and H.3.

6 Experiments

In this section, we evaluate our methods for unconditional and constrained tabular generation on eight datasets: Shoppers, Magic, Default, News, Beijing, and Adult from Zhang et al. (2024); Diabetes from Shi et al. (2025); and the larger FB-comments from Gorishniy et al. (2021). We use published splits (Appendix I).

Table 2: **Unconditional generation on six largest datasets.** We report TREND error (%), C2ST-XGB, MLE (classification AUC, regression RMSE†), and DCR balance. Underlined: better of original and symmetric variants; **blue**: best across models. ↓ lower is better, ↑ higher is better. Other datasets and metrics in Appendix K.1.

	TABDIFF		TABBYFLOW		TABPC	TABJUMP	TABDIFF		TABBYFLOW		TABPC	TABJUMP
	orig.	*sym*	orig.	*sym*			orig.	*sym*	orig.	*sym*		
	TREND error (%) ↓						**C2ST-XGB ↑**					
DEFAULT	4.61	3.61	5.08	**2.13**	5.56	2.72	0.60	0.72	0.51	0.46	0.27	**0.75**
NEWS	1.85	**1.06**	1.60	1.18	1.87	1.29	0.03	0.32	0.13	0.24	0.12	**0.57**
BEIJING	2.56	2.41	2.23	**1.79**	2.13	1.95	0.78	0.83	0.83	0.74	0.57	**0.95**
ADULT	1.60	**1.22**	1.76	1.23	1.41	1.80	0.84	0.88	0.85	0.87	0.87	**1.00**
DIABETES	2.53	**1.63**	4.20	2.59	1.85	1.90	0.73	0.99	0.49	0.70	0.79	**1.00**
FB-COMMENTS	2.07	2.25	**1.27**	1.44	1.84	1.34	0.61	0.65	0.10	0.17	0.00	**0.84**
	MLE (AUC / RMSE†)						**DCR balance ↓**					
DEFAULT	0.759	0.762	0.752	**0.763**	0.748	0.762	**0.81**	1.00	0.81	1.34	1.10	1.20
NEWS †	0.862	0.855	0.852	**0.846**	0.877	0.853	13.12	10.89	12.34	**10.64**	12.09	11.11
BEIJING †	0.562	0.575	**0.535**	0.605	0.655	0.583	1.78	1.90	3.92	**1.62**	1.94	2.43
ADULT	0.913	0.913	0.911	0.912	0.915	**0.923**	4.00	3.41	6.21	**0.41**	2.61	5.82
DIABETES	0.693	**0.695**	0.693	0.694	0.691	0.695	3.97	3.04	5.82	3.79	**2.99**	3.12
FB-COMMENTS †	0.471	0.472	**0.466**	0.473	0.545	0.482	1.04	1.39	**0.33**	0.93	0.94	0.48

6.1 EVALUATING UNCONDITIONAL TABULAR GENERATION

Setup. We assess whether our column-symmetric core improves generation quality by comparing TABDIFF and TABBYFLOW with their symmetric variants, TABDIFF-*sym* and TABBYFLOW-*sym*, which differ only in the network core (Appendix G.3). We also evaluate TABJUMP and the order-invariant probabilistic circuit TABPC (Scassola et al., 2026). Results for TABSYN and TABSYNFLOW, including their symmetric variants, appear in Appendix K.1. We retain published hyperparameters for existing models and report ours in Appendix G.

Metrics. Following Jiang et al. (2026) and Scassola et al. (2026), we evaluate three axes of generation quality. *Fidelity* includes SHAPE, TREND, C2ST with an XGBoost discriminator (C2ST-XGB), α-precision, and β-recall. *Utility* is machine-learning efficacy (MLE), measured by AUC or RMSE on real data using models trained on synthetic data. *Privacy preservation* includes distance to the closest record (DCR) balance, and exact-match rate (EMR). Appendix J.1 discusses the choice of metrics.

Results. In Table 2, we observe: **(1)** Symmetric-core models attain the best fidelity and utility scores in 15 of 18 entries, including one tie. Notably, TABJUMP achieves the highest C2ST-XGB on all six datasets. **(2)** TABPC outperforms the original TABDIFF on four datasets in TREND and three in C2ST-XGB, consistent with Scassola et al. (2026). TABDIFF-*sym* reverses this on four and six datasets, respectively. **(3)** TABPC remains competitive in TREND, but scores 0.12 and 0.00 in C2ST-XGB on NEWS and FB-COMMENTS. Its pairwise-derived structure may miss higher-order dependencies detected by XGBoost. **(4)** Classification MLE varies by at most 0.02 AUC. Symmetric-core models lead on four datasets; TABPC has the highest RMSE on all regression tasks. **(5)** DCR balance shows no significant outliers, suggesting no privacy cost to fidelity gains.

6.2 EVALUATING CONSTRAINED TABULAR GENERATION

Setup. We condition TABDIFF-*sym* and TABJUMP using our learned guide (§5), and TABBYFLOW-*sym* using the closed-form conditioner (§4.3). We compare against DIFFPUTER (Zhang et al., 2025), HARPOON (Shankar et al., 2026), and GREAT (Borisov et al., 2022). For each dataset, we construct product constraints spanning selectivities, arities, and column types, with an $80/20$ training/test split. For each unseen constraint, we generate $1{,}000$ rows per seed across five seeds and compare them with real rows satisfying it. Figure 2 reports hit rate (constraint satisfaction), SHAPE (marginal fidelity), and C2ST-XGB (joint fidelity) across selectivities. Appendices I.4, J.3, and K.2 give baseline adaptations, the evaluation protocol, and additional results.

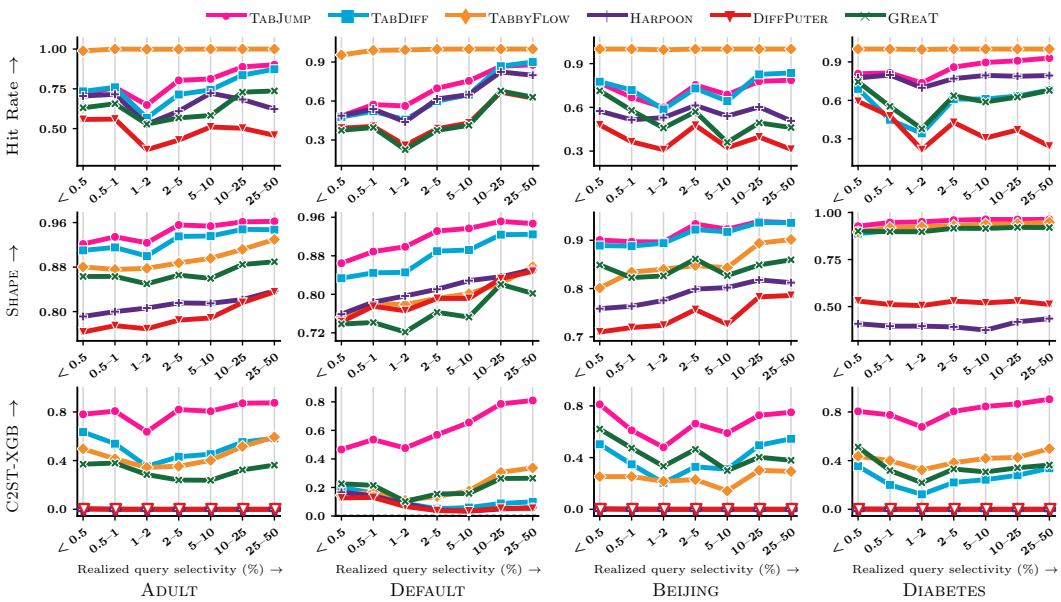


Figure 2: Conditional generation across query selectivity (rows: mean Hit Rate, SHAPE, C2ST-XGB). Hollow markers in the last row denote scores at or below 0.005 (dashed line).

Results. From Figure 2, we observe: **(1)** TABBYFLOW-*sym* achieves near-perfect hit rate across datasets and selectivities, since its analytic conditioner enforces constraints by construction. TABJUMP and TABDIFF-*sym* follow, with TABJUMP generally ahead, possibly because it learns a single guide rather than TABDIFF-*sym*'s two potentially inconsistent corrections. **(2)** Despite its higher hit rate, TABBYFLOW-*sym* generally trails the guided models in SHAPE and TABJUMP in C2ST-XGB, reflecting its biased endpoint posterior (§4.3). **(3)** Both guided models achieve high, comparable SHAPE, but TABJUMP substantially outperforms TABDIFF-*sym* in C2ST-XGB, extending its unconditional advantage, particularly on DEFAULT and DIABETES. **(4)** HARPOON generally achieves the highest hit rate among baselines by directly penalizing constraint violations, but yields poor SHAPE and near-zero C2ST-XGB, consistent with samples concentrating near constraint boundaries. DIFFPUTER also performs poorly, highlighting the limitations of imputation for range constraints. **(5)** GREAT is more competitive in C2ST-XGB, particularly where TABDIFF-*sym* struggles, but trails TABJUMP across all three metrics.

6.3 DEVIATIONS BETWEEN THEORY AND EXPERIMENTS

We discuss three practical limitations that in part may help account for deviations between our theory and experimental results. **(1)** Proposition 4.5 assumes that X is the time reversal of a forward noising process. Since noising reaches stationarity only asymptotically, finite-time noising leaves residual dependence between X_0 and X_1. **(2)** Our finite-capacity models amortize conditioning over many constraint sets. Although the objectives are consistent in the infinite-capacity limit, imperfect learning introduces errors. Moreover, guidance diverges as $t \to 1$, so bounding its strength weakens the guarantees. **(3)** Data quantization and finite-step simulation of the stochastic process introduce further errors. In high-dimensional tables, particularly for rare constraints, these errors can accumulate and lead to constraint violations during stochastic sampling.

7 CONCLUSION

We unify tabular generative models through Markov generators, and motivate TABJUMP, a new pure-jump model. Using Doob's h-transform, we derive exact generator corrections that guarantee sampling from the true conditional distribution and characterize processes amenable to conditioning. We propose a scalable, model-agnostic framework with a consistent learning objective, amortizing conditioning across large constraint sets. Across eight datasets, our methods improve unconditional and conditional fidelity. Our framework extends to other mixed-type domains, with coupled processes admitting simultaneous jumps as a promising direction for future work.

AI USE STATEMENT

We used AI tools for writing assistance, experimental code and plots, and reviewing related work to evaluate the novelty and positioning of our contributions. The proofs were largely developed by the authors, with limited AI assistance for selected propositions, checking arguments, and improving exposition. All proofs and paper contents were verified by the authors, who take full responsibility for the correctness of the manuscript.

ETHICS STATEMENT

We use publicly available codebases and datasets and do not foresee any ethical considerations of this work.

REPRODUCIBILITY STATEMENT

We provide anonymized code covering all model variants, conditioning methods, evaluation pipelines, and tests, with configurations and dependencies specified. We use public datasets and their published splits, and report results over 20 (unconditional) and 5 (conditional) sampling seeds per model, with a single training seed. The appendix states the assumptions underlying our theoretical results and provides complete proofs of all claims. It also details the training protocols, baseline adaptations, metrics, and evaluation procedures. All evaluation seeds are fixed to ensure reproducibility. Anonymized code is available at `https://anonymous.4open.science/r/ICLR-constrained-tabular-generation`.

REFERENCES

Ahmed Alaa, Boris Van Breugel, Evgeny S Saveliev, and Mihaela Van Der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *International conference on machine learning*, pp. 290–306. PMLR, 2022.

Robert B. Ash and Catherine Doléans-Dade. *Probability and Measure Theory*. Harcourt/Academic Press, San Diego, CA, 2nd edition, 2000. ISBN 9780120652020. URL `https://books.google.fi/books?id=GkqQoRpCO2QC`.

Paolo Baldi. *Stochastic Calculus: An Introduction Through Theory and Exercises*. 01 2017. ISBN 978-3-319-62225-5. doi: 10.1007/978-3-319-62226-2.

Andrea Bertazzi, Dario Shariatian, Umut Simsekli, Eric Moulines, and Alain Durmus. Piecewise deterministic generative models. *Advances in Neural Information Processing Systems*, 37:93336–93373, 2024.

Carsten Binnig, Donald Kossmann, Eric Lo, and M Tamer Özsu. Qagen: generating query-aware test databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pp. 341–352, 2007.

Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280*, 2022.

Andrew Campbell, William Harvey, Christian Weilbach, Valentin De Bortoli, Thomas Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models. *Advances in Neural Information Processing Systems*, 36:42217–42257, 2023.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 2024.

Mihaela Cătălina Stoian, Eleonora Giunchiglia, and Thomas Lukasiewicz. A survey on tabular data generation: Utility, alignment, fidelity, privacy, and beyond. *arXiv e-prints*, pp. arXiv–2503, 2025.

Haotian Chen, Yiting Shen, Jichun Li, and Weizhong Zhao. An effective fragment-based dual conditional diffusion framework for molecular generation. *Briefings in Bioinformatics*, 27(1): bbaf727, 2026.

Chaoran Cheng, Boran Han, Danielle Maddix, Abdul Fatir Ansari, Andrew Stuart, Michael W Mahoney, and Bernie Wang. Gradient-free generation for hard-constrained systems. In *International Conference on Learning Representations*, volume 2025, pp. 100510–100539, 2025.

Chaoran Cheng, Jiaqi Guan, Milong Ren, Chengyue Gong, Cong Liu, Xinshi Chen, Ge Liu, and Wenzhi Xiao. A-code: Fully atomic protein co-design with unified multimodal diffusion. *arXiv preprint arXiv:2605.03360*, 2026.

Jacob K Christopher, Stephen Baek, and Ferdinando Fioretto. Constrained synthesis with projected diffusion models, 2024. *URL https://arxiv. org/abs/2402.03559*.

Marc Corstanje, Frank van der Meulen, and Moritz Schauer. Conditioning continuous-time markov processes by guiding. *Stochastics*, 95(6):963–996, 2023.

Philippe Courrège. Sur la forme intégro-différentielle des opérateurs de c_k^∞ dans c satisfaisant au principe du maximum. *Séminaire de théorie du potentiel*, 10(1):1–38, 1965-1966. URL `https://www.numdam.org/item/SBCD_1965-1966__10_1_A2_0/`. talk:2.

Synthetic Data Metrics. DataCebo, Inc., 10 2025. URL `https://docs.sdv.dev/sdmetrics/`. Version 0.24.0.

Alexander Denker, Francisco Vargas, Shreyas Padhy, Kieran Didi, Simon Mathis, Vincent Dutordoir, Riccardo Barbano, Emile Mathieu, Urszula J Komorowska, and Pietro Lio. Deft: Efficient fine-tuning of diffusion models by learning the generalised h-transform. *Advances in Neural Information Processing Systems*, 37:19636–19682, 2024.

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

Kieran Didi, Francisco Vargas, Simon V Mathis, Vincent Dutordoir, Emile Mathieu, Urszula J Komorowska, and Pietro Lio. A framework for conditional diffusion modelling with applications in motif scaffolding for protein design. *arXiv preprint arXiv:2312.09236*, 2023.

Joseph L Doob. Conditional brownian motion and the boundary limits of harmonic functions. *Bulletin de la Société mathématique de France*, 85:431–458, 1957.

Yuanqi Du, Michael Plainer, Rob Brekelmans, Chenru Duan, Frank Noe, Carla P Gomes, Alan Aspuru-Guzik, and Kirill Neklyudov. Doob's lagrangian: A sample-efficient variational approach to transition path sampling. *Advances in Neural Information Processing Systems*, 37:65791–65822, 2024.

Floor Eijkelboom, Grigory Bartosh, Christian A Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. *Advances in Neural Information Processing Systems*, 37:11735–11764, 2024.

Floor Eijkelboom, Heiko Zimmermann, Sharvaree Vadgama, Erik J. Bekkers, Max Welling, Christian A. Naesseth, and Jan-Willem van de Meent. Controlled generation with equivariant variational flow matching. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL `https://proceedings.mlr.press/v267/eijkelboom25a.html`.

Justin Engelmann and Stefan Lessmann. Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning. *Expert Systems with applications*, 174:114582, 2021.

S.N. Ethier and T.G. Kurtz. *Markov Processes: Characterization and Convergence*. Wiley Series in Probability and Statistics. Wiley, 2009. ISBN 9780470317327. URL `https://books.google.fi/books?id=zvE9RFouKoMC`.

Zhaoxing Gan and Guangnan Ye. Doglayout: Denoising diffusion gan for discrete and continuous layout generation. *arXiv preprint arXiv:2412.00381*, 2024.

Igor Vladimirovich Girsanov. On transforming a certain class of stochastic processes by absolutely continuous substitution of measures. *Theory of Probability & Its Applications*, 5(3):285–301, 1960.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943, 2021.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004, 2022.

Zhengyi Guo, Wenpin Tang, and Renyuan Xu. Conditional diffusion guidance under hard constraint: A stochastic analysis approach. *arXiv preprint arXiv:2602.05533*, 2026.

Andrés Guzmán-Cordero, Floor Eijkelboom, and Jan-Willem Van De Meent. Exponential family variational flow matching for tabular data generation. *arXiv preprint arXiv:2506.05940*, 2025.

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

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

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky TQ Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. In *International Conference on Learning Representations*, volume 2025, pp. 52153–52219, 2025.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.

Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Layoutdm: Discrete diffusion model for controllable layout generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10167–10176. IEEE, 2023.

Xiangjian Jiang, Nikola Simidjievski, and Mateja Jamnik. Tabstruct: Measuring structural fidelity of tabular data. In *International Conference on Learning Representations*, volume 2026, pp. 41147–41209, 2026.

Rui Jiao, Wenbing Huang, Peijia Lin, Jiaqi Han, Pin Chen, Yutong Lu, and Yang Liu. Crystal structure prediction by joint equivariant diffusion. *Advances in Neural Information Processing Systems*, 36:17464–17497, 2023.

Wonsuhk Jung, Utkarsh A Mishra, Nadun Ranawaka Arachchige, Yongxin Chen, Danfei Xu, and Shreyas Kousik. Joint model-based model-free diffusion for planning with constraints. *arXiv preprint arXiv:2509.08775*, 2025.

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

Subhojyoti Khastagir, Kishalay Das, Pawan Goyal, Seung-Cheol Lee, Satadeep Bhattacharjee, et al. Llm meets diffusion: a hybrid framework for crystal material generation. *Advances in Neural Information Processing Systems*, 38:169190–169215, 2026.

Minwoo Kim, Seunghyeok Shin, and Hongki Lim. Fast-dips: Adjoint-free analytic steps and hard-constrained likelihood correction for diffusion-prior inverse problems. *arXiv preprint arXiv:2603.01591*, 2026.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In *International conference on machine learning*, pp. 17564–17579. PMLR, 2023.

Bogdan Kulynych, Theresa Stadler, Jean Louis Raisaro, and Carmela Troncoso. Should i use synthetic data for that? an analysis of the suitability of synthetic data for data sharing and augmentation. *arXiv preprint arXiv:2602.03791*, 2026.

Maxime Langevin, Hervé Minoux, Maximilien Levesque, and Marc Bianciotto. Scaffold-constrained molecular generation. *Journal of chemical information and modeling*, 60(12):5637–5646, 2020.

Elad Levi, Eli Brosh, Mykola Mykhailych, and Meir Perez. Dlt: Conditioned layout generation with joint discrete-continuous diffusion layout transformer. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2106–2115. IEEE, 2023.

Ruoyan Li, Dipti Ranjan Sahu, Guy Van den Broeck, and Zhe Zeng. Deep generative models with hard linear equality constraints. *arXiv preprint arXiv:2502.05416*, 2025.

Jinhao Liang, Jacob K Christopher, Sven Koenig, and Ferdinando Fioretto. Simultaneous multi-robot motion planning with projected diffusion models. *arXiv preprint arXiv:2502.03607*, 2025.

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

Ke Liu, Weian Mao, Shuaike Shen, Xiaoran Jiao, Zheng Sun, Hao Chen, and Chunhua Shen. Floating anchor diffusion model for multi-motif scaffolding. *arXiv preprint arXiv:2406.03141*, 2024.

Eric Lo, Nick Cheng, and Wing-Kai Hon. Generating databases for query workloads. *Proceedings of the VLDB Endowment*, 3(1-2):848–859, 2010.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11451–11461. IEEE, 2022.

Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *arXiv preprint arXiv:2201.12677*, 2022.

Benjamin Kurt Miller, Ricky TQ Chen, Anuroop Sriram, and Brandon M Wood. Flowmm: Generating materials with riemannian flow matching. *arXiv preprint arXiv:2406.04713*, 2024.

Bahrul Ilmi Nasution, Floor Eijkelboom, Mark Elliot, Richard Allmendinger, and Christian A Naesseth. Flow matching for tabular data synthesis. *arXiv preprint arXiv:2512.00698*, 2025.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. In *International Conference on Learning Representations*, volume 2025, pp. 36052–36106, 2025.

Zbigniew Palmowski and Tomasz Rolski. A technique for exponential change of measure for markov processes. 2002.

Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in neural information processing systems*, 34:12013–12026, 2021.

Amnon Pazy. *Semigroups of linear operators and applications to partial differential equations*. Springer Science & Business Media, 2012.

Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*, volume 293 of *Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, third edition, 1999. ISBN 3-540-64325-7. doi: 10.1007/978-3-662-06400-9. URL https://doi.org/10.1007/978-3-662-06400-9.

Rick Sauber-Cole and Taghi M Khoshgoftaar. The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey. *Journal of Big Data*, 9(1):98, 2022.

Davide Scassola, Dylan Ponsford, Adrián Javaloy, Sebastiano Saccani, Luca Bortolussi, Henry Gouk, and Antonio Vergari. A sobering look at tabular data generation via probabilistic circuits. *arXiv preprint arXiv:2603.23016*, 2026.

Kiyoung Seong, Sungsoo Ahn, Sehui Han, and Changyoung Park. Multimodal crystal flow: Any-to-any modality generation for unified crystal modeling. *arXiv preprint arXiv:2602.20210*, 2026.

Aditya Shankar, Yuandou Wang, Rihan Hai, and Lydia Chen. Harpoon: Generalised manifold guidance for conditional tabular diffusion. In *International Conference on Learning Representations*, volume 2026, pp. 143083–143109, 2026.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. Tabdiff: a mixed-type diffusion model for tabular data generation. In *International Conference on Learning Representations*, volume 2025, pp. 37353–37375, 2025.

Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. *Advances in neural information processing systems*, 36:62183–62223, 2023.

Bernard Walter Silverman and Frank S. Kelly. Reversibility and stochastic networks. 1980. URL `https://api.semanticscholar.org/CorpusID:125211322`.

Bowen Song, Soo Min Kwon, Zecheng Zhang, Xinyu Hu, Qing Qu, and Liyue Shen. Solving inverse problems with latent diffusion models via hard data consistency. In *International Conference on Learning Representations*, volume 2024, pp. 7624–7654, 2024.

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

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20507–20518. IEEE, 2024.

Utkarsh Utkarsh, Pengfei Cai, Alan Edelman, Rafael Gomez-Bombarelli, and Christopher Rackauckas. Physics-constrained flow matching: Sampling generative models with hard constraints. *Advances in Neural Information Processing Systems*, 38:160217–160252, 2026.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Clement Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. Midi: Mixed graph and 3d denoising diffusion for molecule generation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 560–576. Springer, 2023.

Wilhelm von Waldenfels. Fast positive operatoren. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 4(2):159–174, 1965.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023.

Sen Yang, Lingli Ju, Peng Cheng, JiangLin Zhou, Yamin Cai, and Dawei Feng. Co-design protein sequence and structure in discrete space via generative flow. *Bioinformatics*, 41(5):btaf248, 2025.

Mao Ye, Lemeng Wu, and Qiang Liu. First hitting diffusion models for generating manifold, graph and categorical data. *Advances in Neural Information Processing Systems*, 35:27280–27292, 2022.

Eun Bi Yoon, Keehun Park, Sungwoong Kim, and Sungbin Lim. Score-based generative models with lévy processes. *Advances in neural information processing systems*, 36:40694–40707, 2023.

Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Zilong Wang, Aliaksandra Shysheya, Jonathan Crabbé, Shoko Ueda, et al. A generative model for inorganic materials design. *Nature*, 639(8055):624–632, 2025.

Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *International Conference on Learning Representations*, volume 2024, pp. 52829–52857, 2024.

Hengrui Zhang, Liancheng Fang, Qitian Wu, and Philip Yu. Diffputer: Empowering diffusion models for missing data imputation. In *International Conference on Learning Representations*, volume 2025, pp. 63164–63185, 2025.

Shuhan Zheng and Nontawat Charoenphakdee. Diffusion models for missing value imputation in tabular data. *arXiv preprint arXiv:2210.17128*, 2022.

Constrained Tabular Generation: Exact Conditioning of Arbitrary Markov Processes (Appendix)

CONTENTS

A Related Work **18**

A.1 Mixed-Type Generation and Exact Conditioning beyond Tables 18

A.2 Doob's H-Transform 19

B Setup and Notation **20**

B.1 Definitions and Assumptions 20

B.2 Markov Processes 20

B.3 Feller Processes 21

B.4 Regularity Assumptions 22

C Derivation of generators **23**

D Markovian Projections **24**

E Proofs **25**

E.1 Proof of Theorem 4.1 25

E.2 Proof of Theorem 4.2 28

E.3 Proof of Corollary 4.3 31

E.3.1 Flows 31

E.3.2 Diffusions 32

E.3.3 Jumps 32

E.4 Proof of Corollary 4.4 32

E.5 Proof of Proposition 4.5 33

E.6 Proofs for training matching 34

E.7 Proof of Consistency 35

E.8 Necessity of Assumption 3 of B.4 35

F Explicit examples of conditioning **36**

F.1 Closed-form conditioner for TABBYFLOW 36

G Unconditional models: description and details **37**

G.1 Notation, preprocessing, and time 37

G.2 Baselines and settings 37

G.3 The symmetric core: architecture and column symmetry 38

G.3.1 Architecture and hyperparameters 38

G.3.2 Column symmetry: definition and proofs 39

G.4 TABJUMP 41
G.5 Tabular generative models as instances of generator matching 42
G.5.1 TABJUMP as an instance of generator matching 43

H Amortized conditioning: module, losses, training, and sampling **45**
H.1 Architecture 45
H.2 Losses 46
H.3 Training 48
H.4 Sampling 49
H.5 Cost 49

I Datasets, splits, and constraint sets **50**
I.1 Datasets 50
I.2 Dataset splits for unconditional training 51
I.3 Constraint sets for training and evaluating conditioning 51
I.4 Constrained generation baselines 53

J Experimental Methodology **54**
J.1 A Discussion on Metrics Evaluated 54
J.1.1 Fidelity Metrics 54
J.1.2 Privacy Metrics 54
J.1.3 Utility Metrics 55
J.2 Protocol for Unconditional Evaluation 55
J.3 Protocol for Conditional Evaluation 55

K Detailed Experimental Results **55**
K.1 Unconditional Evaluation 55
K.2 Conditional Evaluation 56

A Related Work

A.1 Mixed-Type Generation and Exact Conditioning beyond Tables

Mixed discrete–continuous generation and exact conditioning are not specific to tables. Both arise in several structured generation problems. We briefly discuss these connections and explain why tables provide a useful setting for developing and evaluating the two together.

Mixed-type generation. Several structured generation problems combine discrete and continuous variables. A molecule contains categorical atom types together with continuous 3D coordinates, and molecular generative models therefore combine discrete dynamics over atom types with diffusion or flow over coordinates (Hoogeboom et al., 2022; Vignac et al., 2023; 2022). Similarly, proteins combine a discrete residue sequence with a continuous backbone, with recent sequence–structure models using a continuous-time Markov chain for the sequence and a continuous flow for the structure (Campbell et al., 2024; Cheng et al., 2026; Yang et al., 2025). Crystals combine discrete atomic species with continuous coordinates and lattice parameters (Zeni et al., 2025; Jiao et al., 2023; Miller et al., 2024; Seong et al., 2026; Khastagir et al., 2026). Related mixed-type state spaces also arise in graphic layouts, which combine element categories with bounding boxes (Levi et al., 2023; Inoue et al., 2023; Gan & Ye, 2024), and indoor scenes, which combine object types with continuous poses (Paschalidou et al., 2021; Tang et al., 2024).

These examples have the same basic structure as our setting, where a Markov process evolves on a product of discrete and continuous state spaces. In the terminology of Theorem 3.1, the discrete variables evolve through jump components (CMTCs), while the continuous variables may evolve through flow, diffusion or jump components. The corresponding models can therefore be viewed through the same generator-matching framework used in §3. They also exhibit a related permutation symmetry. Atoms, residues, or scene objects form unordered collections of typed entities, just as a table row can be viewed as the set of typed pairs $(\tau_j, v_j)_j$. Architectures in these domains typically respect this symmetry by construction. Tabular generators, as §3 shows, mostly do not.

Exact conditioning. Hard conditioning also appears in several structured generation problems. One common case is *pinned values*, where part of the generated object must be fixed exactly. In protein design, a functional motif may be fixed while the surrounding scaffold is generated, with approaches based on the Doob h-transform (Didi et al., 2023) or exact terminal feasibility (Christopher et al.; Liu et al., 2024; Watson et al., 2023). Molecular design similarly conditions on a prescribed scaffold or fragment (Langevin et al., 2020; Chen et al., 2026), while inverse problems in imaging condition on observed measurements (Lugmayr et al., 2022; Song et al., 2024; Kim et al., 2026). These correspond to equality constraints on a subset of variables, analogous to imputation in tables.

A second case is *set-valued conditioning*. Robotic trajectories may be required to remain inside a feasible set (Liang et al., 2025; Jung et al., 2025), while safety-critical or rare-event generation may require a specified event to occur with probability one (Guo et al., 2026). These are analogous to the range and set-membership constraints considered in this work. More general functional constraints also arise, for example when generated fields must satisfy conservation laws or boundary conditions (Cheng et al., 2025; Utkarsh et al., 2026; Li et al., 2025). Such constraints still define measurable events, so the theory of §4 applies in principle, although estimating the corresponding hitting probability h may be substantially harder. Across these domains the prevailing practice is soft guidance or post-hoc projection. Existing exact-conditioning methods have largely been developed only for continuous diffusions, which is the gap that §4 closes for arbitrary Markov processes.

Subset conditioning. A related problem is to sample from a designated subpopulation of a pretrained model rather than from its full distribution. Examples include molecular graphs drawn from the neighborhood of a small set of reference molecules, or layouts belonging to a particular style within a general layout model. The target can again be written as $p_{\mathrm{data}}(\cdot \mid S)$ for a low-probability set S. The difficulty is therefore the same as for low-selectivity constraints: rejection sampling becomes expensive, while soft guidance need not recover the conditional distribution. Exact approaches such as twisted sequential Monte Carlo (Wu et al., 2023) and Doob-based rare-event sampling (Du et al., 2024) are again formulated for continuous diffusions. Low-selectivity constraint sets on tables provide a controlled version of this problem.

Why tables. Tables combine all of these challenges: heterogeneous variables, strong cross-column dependencies, and constraints over arbitrary subsets of columns. They also provide two useful properties for evaluating exact conditioning. First, the target conditional distribution is directly observable. Applying `SELECT * WHERE` S to the real data returns the rows satisfying S, allowing generated samples to be compared directly against the corresponding real subpopulation. In domains such as molecules or proteins, the true conditional distribution is generally unavailable and evaluation instead relies on task-specific proxies.

Second, the difficulty of conditioning can be varied systematically. The selectivity $p_{\mathrm{data}}(S)$ can be changed by tightening numerical ranges or categorical membership sets, while the target distribution remains directly measurable from the data. This allows conditioning methods to be evaluated across a controlled range of constraint types and selectivities. We therefore use tables as the main setting for developing and evaluating our framework, while the underlying mixed-type and exact-conditioning problems extend more broadly.

A.2 Doob's h-Transform

A function $h(t, x)$ which is *space-time harmonic* with respect to $\mathcal{L}$, i.e., satisfies $\mathcal{L}h = (\mathcal{L}_t + \partial_t)h = 0$ can be used to change the underlying path measure $\mathbb{P}$ of the markov process. The new measure $\mathbb{Q}$ restricted to an intermediate filtration $\mathcal{F}_t$ is defined via the Radon Nikodym theorem such that $\frac{d\mathbb{Q}}{d\mathbb{P}}\Big|_{\mathcal{F}_t} = h(t, x)/h(0, X_0)$. Applications of doob h-transforms have been seen popularly in literature Palmowski & Rolski (2002), Doob (1957), Girsanov (1960). Our use of the doob's h-transform defines $h(t, x) = p(t, 1, x, S) = \mathbb{P}(X_1 \in S \mid X_t = x)$, the posterior of observing S at terminal time T given we observe x at t. Prior work Guo et al. (2026) has studied this under probability theoretic means however they restrict themselves to diffusions. Other works Du et al. (2024), consider $\nabla h/h$ as the stochastic optimal control to be learned, which is inherently *on policy*. The characterization we provide is inherently on policy and requires no simulation. Further, we highlight that there are plenty of other applications of the doob's transforms by choosing other such space-time harmonic functions. One such applicationis its use in First Hitting Diffusion Models Ye et al. (2022).

We now provide some prior works in which concepts related to the Doob's h-transform have been utilized.

(Corstanje et al., 2023) considers a similar problem of conditioning arbitrary Markov processes beyond diffusion. They prove a theorem similar to Theorem 4.1 for a more general class of functions on the extended generator Palmowski & Rolski (2002). However, there is a large distinction between our approaches. Their goal is to use a fixed auxiliary process which is close to the true h-transform. As such, they do not concern themselves with regularity of the transformed process. Our goal is to learn a neural approximation to h and so we require the Feller property to pass through to the conditioned process X^S.

(Denker et al., 2024) propose finetuning a pretrained diffusion model with the additional h-transform network. They restrict themselves to diffusion however. Our characterization suitably generalizes this, with their loss function obtainable as a special case of the conditioned generator matching framework of Proposition 4.5. We encourage readers to take a look at Appendix **G.2** from their paper to understand the approximation incurred in assuming finite mixing time of the forward noising process (Appendix D).

(Guo et al., 2026) also restrict themselves to diffusions. They, however provide a suitable way to learn h even in the absence of training data by drawing samples from the frozen backbone. In tail events, their characterization is very useful, since they are able to learn from negative samples produced by the unconditional backbone as well.

(Song et al., 2020, Appendix. I2), **Campbell et al. (2023)** and **Watson et al. (2023)** only consider the case of imputation. In this case, they turn to an approximation of the Doob's transform by fixing the coordinates which they observe forcefully, and at each denoising step, fill in the observed coordinates with a noisy version of the input. We note that this is a biased estimator. While the machinery of Doob still applies, the analysis is messy since fixing coordinates yields a measure zero event.

(Nisonoff et al., 2025) look at conditioning in discrete state spaces. They utilize the fact that networks tractably parametrize transitions of the sequence by changing a single coordinate at a time. This is precisely generalized by the multimodal characterizations of generators. Further, their

B Setup and Notation

We give a brief description of markov processes through a measure theoretic lens, so further proofs and properties remain consistent. For interested readers, we recommend reading Ethier & Kurtz (2009). Our exposition follows from Baldi (2017)[Chapter 6], Revuz & Yor (1999)[Chapter 3] and in the same structural manner as Holderrieth et al. (2025)

B.1 Definitions and Assumptions

Observable Space: We always assume that our observable space E is a Polish metric space, that is, there exists a metric $d : E \times E \to \mathbb{R}$ which is complete in its metric topology (any cauchy sequence converges), and that E is separable (admits a countable dense subset). For all intents and purposes of a practitioner, all spaces of interest are polish. We assume that the space is endowed with a σ-algebra $\mathscr{E}$ which in most cases is the borel sigma algebra $\mathscr{B}(E)$ on E. All functions $f : E \to \mathbb{R}$ in question will be measurable.

B.2 Markov Processes

Let us work with an observable space $E, \mathscr{E}$.

Definition B.1 (Markov Transition function). A markov transition function on $(E, \mathscr{E})$ is a function $p(s, t, x, A)$ where $s, t \in \mathbb{R}^+$, $s \leq t$, $x \in E$, $A \in \mathscr{E}$ such that

1. for fixed s, t, A, $x \mapsto p(s, t, x, A)$ is $\mathscr{E}$-measurable;
2. for fixed s, t, x, $A \mapsto p(s, t, x, A)$ is a probability on $(E, \mathscr{E})$
3. p satisfies the *Chapman-Kolmogorov equation*

$$p(s, t, x, A) = \int_E p(u, t, y, A)p(s, u, x, dy) \tag{15}$$

for every $s \leq u \leq t$

Definition B.2 (Markov Process). Given on $(E, \mathscr{E})$ a Markov transition function p and a probability μ, an $(E, \mathscr{E})$-valued process $X = (\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq u}, (X_t)_{t\geq u}, \mathbb{P})$ is said to be a *Markov Process* associated to p, starting at time u and with initial distribution μ, if

1. X_u has law μ on E;
2. *(Markov Property)* For every bounded measurable function $f : E \to \mathbb{R}$ and $t \geq s \geq u$

$$\mathbb{E}[f(X_t) \mid \mathcal{F}_s] = \int_E f(x)p(s, t, X_s, dx) \quad \text{a.s.} \tag{16}$$

Corollary B.3.

$$\mathbb{P}(X_t \in A \mid \mathcal{F}_s) = \mathbb{P}(X_t \in A \mid X_s), \quad \mathbb{E}[f(X_t) \mid \mathcal{F}_s] = \mathbb{E}[f(X_t) \mid X_s] \tag{17}$$

Proof. Choosing $f = \mathbf{1}_A$, $A \in \mathscr{E}$, from 16, we have that $\mathbb{P}(X_t \in A \mid \mathcal{F}_s) = p(s, t, X_s, A)$. Now, by tower property of filtrations, we have that $\mathbb{P}(X_t \in A \mid X_s) = \mathbb{E}[\mathbb{P}(X_t \in A \mid \mathcal{F}_s) \mid X_s]$. Now, since $\mathbb{P}(X_t \in A \mid \mathcal{F}_s)$ is a deterministic function of X_s, we have that $\mathbb{E}[\mathbb{P}(X_t \in A \mid \mathcal{F}_s) \mid X_s] = \mathbb{P}(X_t \in A \mid \mathcal{F}_s) = \mathbb{P}(X_t \in A \mid X_s)$. Thus, we also have that $\mathbb{E}[f(X_t) \mid \mathcal{F}_s] = \mathbb{E}[f(X_t) \mid X_s]$ ■

Marginal Operators: For ease of notation, we define the marginal operator $\langle p_t, . \rangle$ acting on the set of measurable functions $f : E \to \mathbb{R}$ with finite expectations defined as

$$\langle p_t, f \rangle = \mathbb{E}[f(X_t)] = \int_\Omega f(X_t(\omega))d\mathbb{P}(\omega) \tag{18}$$

Semigroups: Let $\mathcal{C}_0(E)$ denote the set of continuous functions $f : E \to \mathbb{R}$ which vanish at infinity. Let p defined above be a transition function, on space E. To p, we can associate the family of linear operators $(T_{s,t})_{s \leq t}$ defined on $\mathcal{C}_0(E)$ by

$$T_{s,t}f(x) = \int_E f(y)p(s,t,x,dy) \tag{19}$$

Further, from *Chapman Kolmogorov*, we obtain for $s \leq u \leq t$, $T_{s,t} = T_{s,u}T_{u,t}$. We also have that $T_{t,t} = Id$. Thus, the family of operators $T_{s,t}$ form a *semigroup* under composition. We further note that $T_{s,t}f(x) = \mathbb{E}[f(X_t) \mid X_s = x] \leq \|f\|_\infty$. Hence $\|T_{s,t}f\|_\infty \leq \|f\|_\infty$, and so the operators are contractions on $C_0(E)$.

Remark B.4. Whenever we define an operator $\mathcal{L}$ on a domain $\mathcal{D}$, we assume that its action $\mathcal{L}f$ on $f \in \mathcal{D}$ also lies in $\mathcal{D}$.

B.3 Feller Processes

A markov transition function p is called time-homogeneous if

$$p(s, s+h, x, A) = p(t, t+h, x, A) \, \forall s, t \geq u \tag{20}$$

Definition B.5 (Time Homogeneous Markov Processes). A markov process is time-homogeneous if it admits a time homogeneous markov transition function. In such a case, the first parameter can be dropped, such that $\bar{p}(h, x, A) = p(s, s+h, x, A)$, where now $\bar{p}$ is well-defined.

So far, and in most examples for machine learning practitioners, the markov processes used are time inhomogeneous. We can homogenize the process in the style of Holderrieth et al. (2025).
Consider the process $\bar{X}_t = (X_t, t)$ on now the extended observable space $E \times \mathbb{R}^+$. The transition function for this process is naturally given as $\bar{p}(h, (x,t), \bar{A}) = p(t, t+h, x, \pi_E(\bar{A}))$ if $\pi_t(\bar{A}) = t+h$ and 0 otherwise where $\pi_E : E \times \mathbb{R}^+ \to E$ and $\pi_t : E \times \mathbb{R}^+ \to \mathbb{R}^+$, given by $\pi_E : \bar{A} = (A, t) \mapsto A$ and $\pi_t : \bar{A} = (A, t) \mapsto t$. The function $\bar{p}$ is indeed a markov transition function.

Proof. For fixed $(h, (x,t), \bar{A}) \mapsto \bar{p}(h, (x,t), \bar{A})$ is given as the composition of the following maps:

1. $\pi_E : \mathbb{R}^+ \times (E \times \mathbb{R}^+) \times 2^{(E \times \mathbb{R}^+)} \to \mathbb{R}^+ \times (E \times \mathbb{R}^+) \times 2^E$, $(h, (x,t), (A, t+h)) \mapsto (h, (x,t), A)$
2. $f : \mathbb{R}^+ \times (E \times \mathbb{R}^+) \times 2^E \to \mathbb{R}^+ \times \mathbb{R}^+ \times E \times 2^E$, $(h, (x,t), A) \mapsto (t, t+h, x, A)$
3. $p : (t, t+h, x, A) \mapsto p(t, t+h, x, A)$

All of these are measurable, and so their composition is measurable. For the second property, we note that $\int_{E \times \mathbb{R}^+} \bar{p}(h, x, dy) = \int_E p(t, t+h, x, dy) = 1$ where the first equality follows by the definition of $\bar{p}$. *Chapman-Kolmogorov* can be checked similarly. ■

Given these transition functions, we can define new operators on $\bar{T}_h$ on $\mathcal{C}_0(E \times \mathbb{R}^+)$ whose action is given as

$$\bar{T}_h f(x,t) = \int_E f(y, t+h)p(t, t+h, x, dy) = \mathbb{E}[f(X_{t+h}, t+h) \mid X_t = x] \tag{21}$$

Definition B.6 (Feller Process). A time-homogeneous markov process $\bar{X}_t$ (on the above domain) with transition *semigroup* $\bar{T}_h$ is a feller process if for any function $f \in \mathcal{C}_0(E \times \mathbb{R}^+)$

1. $\forall h \geq 0$, we have that $\bar{T}_h f \in \mathcal{C}_0(E \times \mathbb{R}^+)$
2. $\lim_{h \downarrow 0} \|\bar{T}_h f - f\|_\infty = 0$

Given a feller process $\bar{X}_t$ defined above, we can look at the expected rate of change of a function f as time evolves. This motivates the definition of a generator of a markov process

Definition B.7 (Generator). The generator of a time-homogeneous feller process is defined as the linear operator given by the limit

$$\mathcal{L}f := \lim_{h \downarrow 0} \frac{\bar{T}_h f - f}{h} \tag{22}$$

for all $f \in \mathcal{C}_0(E \times \mathbb{R}^+)$ where the limit exists in supremum norm, that is $\lim_{h\downarrow 0} \frac{\|\bar{T}_h f - f - h\mathcal{L}f\|_\infty}{h} = 0$.
We refer to this as $\mathcal{D}(\mathcal{L})$, that is the domain of $\mathcal{L}$
From Holderrieth et al. (2025), we get that the time independent generator acting on $f \in \mathcal{C}_0(E \times \mathbb{R}^+)$

$$\mathcal{L}f(x) = \lim_{h \downarrow 0} \frac{\mathbb{E}[f(X_{t+h}, t+h) - f(X_t, t) \mid X_t = x]}{h} \tag{23}$$

$$= \lim_{h \downarrow 0} \frac{\mathbb{E}[f(X_{t+h}, t+h) - f(X_t, t+h) + f(X_t, t+h) - f(X_t, t) \mid X_t = x]}{h} \tag{24}$$

$$= \lim_{h \downarrow 0} \frac{\mathbb{E}[f(X_{t+h}, t+h) - f(X_t, t+h) + f(X_t, t+h) - f(X_t, t) \mid X_t = x]}{h} \tag{25}$$

$$= \lim_{h \downarrow 0} \frac{\mathbb{E}[f(X_{t+h}, t+h) - f(X_t, t+h) \mid X_t = x] + f(X_t, t+h) - f(X_t, t)}{h} \tag{26}$$

$$\tag{27}$$

Assuming both limits exist, let us define

$$\mathcal{L}_t f^t(x) := \lim_{h \downarrow 0} \frac{\mathbb{E}[f(X_{t+h}, t+h) - f(X_t, t+h) \mid X_t = x]}{h} \tag{28}$$

$$\partial_t f(x,t) := \lim_{h \downarrow 0} \frac{f(x, t+h) - f(x,t)}{h}$$

where $f^t(x) : E \to \mathbb{R}; x \mapsto f(x,t)$. Then $\mathcal{L}f(x,t) = \mathcal{L}_t f^t(x) + \partial_t f(x,t)$. Here, we can define the class of functions $\mathcal{D}(\mathcal{L}_t) \subseteq \mathcal{C}_0(E)$ for which $\mathcal{L}_t$ is well defined. We have that $g \in \mathcal{D}(\mathcal{L}_t)$ if there exists a function $f \in \mathcal{D}(\mathcal{L})$ such that $g(x) = f(x,t)$.
Moreover given some $g \in \mathcal{D}(\mathcal{L})$, if $g(x,s) = g(x,t) \forall s,t$, then we can simply drop the time dependency, and can define

$$\mathcal{L}_t g(x) = \lim_{h \downarrow 0} \frac{\mathbb{E}[g(X_{t+h}) - g(X_t) \mid X_t = x]}{h} \tag{29}$$

Here, the limit exists in supremum norm by hypothesis.

B.4 Regularity Assumptions

We pick up the regularity assumptions as cited in Holderrieth et al. (2025). Further, we ask for more regularity on the transformed process as well. Most markovian processes in practice indeed satisfy them. For intuitions and motivations, we recommend the reader to Appendix [A.2] of Holderrieth et al. (2025). Further, Assuptions $2, 3, 4$ are new which are required for conditioning a process.

1. The Markov Process X_t is feller as defined in B.6
2. The terminal transition function to arbitrary measurable sets S, i.e., $p(s, T, x, S)$ is continuous in (s, x) wherever it is positive, and extends continuously to 0 as $h(s,x) \to 0$ and $h(t,y) \in \mathcal{D}(\mathcal{L}_t) \forall t$
3. For all compact $K \subset U_t$, where $U_t \subset E, x \in U_t \Leftrightarrow h(t,x) > 0$ and $t < T$,

$$\frac{\int_K p(s,t,x,dy) h(t,y)}{\int_{U_t} p(s,t,x,dy) h(t,y)} \to 0 \text{ as } |x| \to \infty \quad \forall s \leq t$$

This assumption is both necessary and sufficient for the transformed process to be feller. The necessity is proved in Section E.8

4. For all compact balls of radius R, $B_R \subset U_t$ and $t < T$,

$$\lim_{t \downarrow s} \sup_{|x|>2R} \frac{1}{t-s} \frac{\int_{B_R} p(s,t,x,dy)h(t,y)}{\int_{U_t} p(s,t,x,dy)h(t,y)} = 0 \quad \forall s \leq t$$

   This is sufficient to show that the generator indeed converges in supremum norm.
5. In every time interval $[s,t]$, the expected number of discontinuities of $t \mapsto X_t(\omega)$ is finite.
6. There is a "nice" class of test functions $\mathcal{T} = \mathcal{C}_c^\infty(E) \subset \bigcap_t \mathcal{D}(\mathcal{L}_t)$ such that $t \mapsto \mathcal{L}_t f$ is continuous for all $f \in \mathcal{T}$. Also, two probability measures μ_1, μ_2 on E are equal a.s. if and only if $\mathbb{E}_{x\sim\mu_1}[f(x)] = \mathbb{E}_{x\sim\mu_2}[f(x)]$ for all $f \in \mathcal{T}$.
7. We assume that the function $t \mapsto \langle p_t, f\rangle$ is continuous in t for all $f \in \mathcal{T}$.

For the regularity assumptions mentioned above, the KFE is indeed well defined. Following is an elementary proof which relies on the following identity Ethier & Kurtz (2009)[Chapter 4, Proposition 1.7], which states that given a markov process X_t with generator $\mathcal{L}_t$, and $f \in \mathcal{D}(\mathcal{L}_t)$,

$$M_t^f = f(X_t) - f(X_0) - \int_0^t \mathcal{L}_s f(X_s) ds \tag{30}$$

is a $\mathcal{F}_t$-martingale. Taking expectations on both sides and applying fubini we get that

$$\langle p_t, f\rangle - \langle p_0, f\rangle - \int_0^t \langle p_s, \mathcal{L}_s f\rangle ds = 0 \tag{31}$$

For a fixed $f \in \mathcal{T}$, let us define $\phi(s) := s \mapsto \mathcal{L}_s f$ and $\psi(s,g) := s \mapsto \langle p_s, g\rangle$, both of which are continuous in s. Then, $\langle p_s, \mathcal{L}_s f\rangle = \psi(s, \phi(s))$. From assumptions 6 and 7, we know that $\phi(s)$ and $\psi(s,g)$ are continuous in s. Thus, $\psi(s,\phi(s))$ is also continuous in s. By the fundamental theorem of calculus, $\langle p_t, f\rangle$ admits a partial derivative with respect to time given as $\langle p_s, \mathcal{L}_s f\rangle$. Thus, the KFE is well defined.

Remark B.8. For the Theorem 3.1, we only require Assumptions 1,5,6,7. Assumptions 2,3,4 are for showing that the the transformed process X^S is still feller.

C Derivation of generators

Our exposition follows from the theory of almost positive operators. Let us fix $E = \mathbb{R}^m \times F$ where F is finite and let $\mathcal{L}$ be the generator of the time-homogeneous markov process on E satisfying the assumptions of Appendix B.4. We assume from Pazy (2012) that $\mathcal{C}_c^\infty(E) \subset \mathcal{D}(\mathcal{L})$. Since F is finite, we can decompose any function $f(x,c) \in \mathcal{C}_c^\infty(E)$ as $f = \sum_{c\in F} f(\cdot,c) \otimes \mathbf{1}_c$, where $(g \otimes \mathbf{1}_c)(y,c') := g(y)\mathbf{1}[c' = c]$. It is a fact that the generator of a markov process is *almost positive*, that is if $f \geq 0$ and $f(x,c) = 0$, then $\mathcal{L}f(x,c) \geq 0$. Since $\mathcal{L}$ is linear, we can decompose the action of $\mathcal{L}$ on f onto its individual components. Let us then evaluate $\mathcal{L}f(x,c)$ on some $c \in F$. We do this in two steps:

Step 1: Restrict the action of $\mathcal{L}$ to $\mathbb{R}^m$ by fixing c. Given c, we can define a new operator $\mathcal{J}$ on $C_c^\infty(\mathbb{R}^m)$ as $\mathcal{J}g(x) = \mathcal{L}[g \otimes \mathbf{1}_c](x,c)$ for $g \in \mathcal{C}_c^\infty(\mathbb{R}^m)$. $\mathcal{J}$ defined this way is also an *almost positive operator* on $\mathbb{R}^m$, and so by von Waldenfels (1965, Satz 1) (see also Courrège, 1965-1966)

$$\mathcal{J}g(x) = -\kappa(x,c)\, g(x) + \nabla g(x)^\top u(x,c) + \tfrac{1}{2}\nabla^2 g(x) \cdot \sigma^2(x,c) + \int_{\mathbb{R}^m} \big[g(y) - g(x)\big]\, Q(\mathrm{d}y, c;\, x, c), \tag{32}$$

with $\kappa \geq 0$, σ^2 positive semi-definite, and $Q(\cdot, c; x, c)$ a measure on $\mathbb{R}^m \setminus \{x\}$; where the extra term of the $\nabla g(x)\mathbf{1}_{\|y-x\|\leq 1}$ in a general representation is removed by the finite-expected jumps assumption, as in Holderrieth et al. (2025). Note here, that $\kappa \neq 0$ in general since the operator may leak some mass to other $c' \neq c$.

Step 2: We now choose some $c' \neq c$. Given some $g \geq 0 \in \mathcal{C}_c^\infty(\mathbb{R}^m)$, we observe that $g \otimes \mathbf{1}_{c'} \geq 0$ and in particular, $[g \otimes \mathbf{1}_{c'}](x,c) = 0$. Then since $\mathcal{L}$ is *almost positive* and $g \otimes \mathbf{1}_{c'} \geq 0$, $[g \otimes \mathbf{1}_{c'}](x,c) = 0 \implies \mathcal{L}[g \otimes \mathbf{1}_{c'}](x,c) = 0$. Hence, $g \mapsto \mathcal{L}[g \otimes \mathbf{1}_{c'}](x,c)$ is a *positive* linear functional on $\mathbb{R}^m$. By the *Riesz-Markov-Kakutani* representation theorem, $\mathcal{L}[g \otimes \mathbf{1}_{c'}](x,c) = \int_{\mathbb{R}^m} g(y) Q(\mathrm{d}y, c'; x, c)$

Now by linearity,

$$\mathcal{L}f(x,c) = \mathcal{J}\big(f(\cdot,c)\big)(x) + \sum_{c'\neq c}\int_{\mathbb{R}^m} f(y,c')\,Q(\mathrm{d}y,c';\,x,c). \tag{33}$$

Applying Eq. 33 to $f \equiv 1$ (since $f \equiv 1$ or any constant function is in $\mathcal{D}(\mathcal{L})$) and using conservativeness, since the process as a whole does not die out, we get that $\kappa(x,c) = \sum_{c'\neq c} Q(\mathbb{R}^m, c'; x, c)$. From Assumption 5 of B.4, this measure is finite. Expanding Eq. 33 gives us that

$$\mathcal{L}f(x,c) = -\kappa(x,c)f(x,c) + \nabla_{\mathrm{num}} f(x,c)^\top u(x,c) + \tfrac{1}{2}\nabla^2_{\mathrm{num}} f(x,c)\cdot\sigma^2(x,c) \tag{34}$$

$$+\int_{\mathbb{R}^m}\big[f(y,c)-f(x,c)\big]\,Q(\mathrm{d}y,c;\,x,c) + \sum_{c'\neq c}\int_{\mathbb{R}^m}\big[f(y,c')\big]\,Q(\mathrm{d}y,c';\,x,c) \tag{35}$$

$$= -\sum_{c'\neq c} Q(\mathbb{R}^m,c';x,c)f(x,c) + \nabla_{\mathrm{num}} f(x,c)^\top u(x,c) + \tfrac{1}{2}\nabla^2_{\mathrm{num}} f(x,c)\cdot\sigma^2(x,c) \tag{36}$$

$$+\int_{\mathbb{R}^m}\big[f(y,c)-f(x,c)\big]\,Q(\mathrm{d}y,c;\,x,c) + \sum_{c'\neq c}\int_{\mathbb{R}^m}\big[f(y,c')\big]\,Q(\mathrm{d}y,c';\,x,c) \tag{37}$$

$$\tag{38}$$

We observe that $\sum_{c'\neq c} Q(\mathbb{R}^m,c';x,c)f(x,c) = \sum_{c'\neq c}\int_{\mathbb{R}^m}\big[f(x,c)\big]\,Q(\mathrm{d}y,c';\,x,c)$
$= \int_{\mathbb{R}^m\times\{c'\neq c\}}\big[f(x,c)\big]\,Q(\mathrm{d}y,\mathrm{d}c';\,x,c)$. Combining, we get that

$$\mathcal{L}f(x,c) = \nabla_{\mathrm{num}} f(x,c)^\top u(x,c) + \tfrac{1}{2}\nabla^2_{\mathrm{num}} f(x,c)\cdot\sigma^2(x,c) \tag{39}$$

$$+\int_E\big[f(y,c')-f(x,c)\big]\,Q(\mathrm{d}y,\mathrm{d}c';\,x,c), \tag{40}$$

where $Q(\cdot\,;x,c)$ is the finite measure on E constructed from the combining the individual Q measures. This is akin to Eq. 4 for a time-homogeneous process.

For a time-inhomogeneous process, we can apply the above to the space-time process (X_t,t) of Appendix B.3; as in Holderrieth et al. (2025, App. C.1.1), which yields Eq. 4 with time-dependent parameters (u_t,σ_t^2,Q_t). (Note that time has unit drift, and no diffusion, or jumps). As special cases, for $|F|=1$ there are no between-slice terms and the result is the Euclidean case. For $m=0$ the differential terms vanish, and the measures form the rate matrix $Q(c',c)$, with $\sum_{c'\neq c}Q(c',c)=0$.

D Markovian Projections

We consider here the conversation from Proposition 4.5. To illustrate an example, let us define the following process which we shall call Y. First, draw $z\sim p_{data}$, then simulate the process with time marginals $p_t(\cdot\mid z)$. First and foremost, the process Y defined this way may not even be Markovian. For a markov process, $\mathbb{P}(X_t\mid\mathcal{F}_s)=\mathbb{P}(X_t\mid X_s)$. As a simple example, as to why this does not hold for Y, consider the CondOT flow model Lipman et al. (2022). Here, if in addition to X_s, the entire path history is given, then the final state X_1 becomes deterministic (since we only need two points for linear interpolation). Meanwhile given only X_s, the endpoint is not deterministic (since we do not know its starting point). Instead, what Holderrieth et al. (2025) suggests is to construct an equivalent markov process which has the same time marginals as Y. Let us call this operation Π, the so-called *projection* and the resulting process $X=\Pi(Y)$ whose generator is defined by $\mathcal{L}_t f(x) = \mathbb{E}_{z\sim p_{1|t}(\cdot|x)}[\mathcal{L}^z_t f(x)]$. Now, the goal of conditioning via the doob's h-transform, is to reweight the existing path measure of X to obtain X^S. For per sample paths ending in S, we can similarly define the process Y^S, to draw $z\sim p^S_{\mathrm{data}}$ and then simulate $p_t(\cdot\mid z)$. The conditional generator matching framework of Proposition 4.5, then aims to learn the parameters of $\Pi(Y^S)$. Now inorder for us to be consistent, we would require that $\Pi(Y^S)=X^S$, that is we would want the diagram below to commute. In the diagram below, $X\mapsto X^S$ refers to transforming a process to hit the constraint set S which we (with some abuse in notation depict with h)

$$\begin{array}{ccc} Y & \xrightarrow{\quad h \quad} & Y^S \\ \Big\downarrow \Pi & & \Big\downarrow \Pi \\ X & \xrightarrow[\quad h \quad]{} & X^S \end{array}$$

This is not true in general. However, a sufficient condition for this to be consistent if the per-sample paths are *bridges* Shi et al. (2023) of some markov process Z which hit terminal value z. Given a path measure $\mathbb{P}$ and a process Y, the path measure decomposes conditioned on the terminal distribution p_{data}, as

$$\mathbb{P}(\cdot) = \int_E \mathbb{P}(\cdot \mid Y_1(\omega) = z) p_{\text{data}}(dz)$$

Note that this holds for any process Y whether it is markov or not. Now, if we consider the process Y to be markov itself, then the above diagram collapses with $\Pi(Y) = Y = X$ as processes, and so we indeed learn a consistent process. This also explains why Proposition 4.5 does not hold for flows. This does hold for diffusions and masked chains, since their per sample paths are indeed bridges of a markov process defined as the time-reversal of the corresponding forward (from $t = 1$ to $t = 0$) noising process. In such a case, the proof is present in Appendix E.5

E PROOFS

E.1 PROOF OF THEOREM 4.1

It is not at all obvious as to why a process with the given generator satisfies the regularity assumptions as stated in B.4. We construct a process, for which the asserted quantity is the generator. Secondly, we also show that the resulting process is Feller and satisfies the regularity conditions as mentioned in Holderrieth et al. (2025). We adapt the proof from Guo et al. (2026).

Remark E.1. Note that, we show this for a general T. All results can be obtained by setting $T = 1$ in the given proof.

Proof. Given the markov process $X = (\Omega, (X_t)_{t\in[0,T]}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P})$ with initial law $p_0(.)$ where $\mathcal{F}_t = \sigma(X_s : s \le t)$ is the canonical filtration. We use $\mathbb{P}$ and $\mathbb{P}_{[0,T]}$ interchangably in the proof.

First, let us define the normalized terminal indicator $Z := \dfrac{\mathbf{1}(X_T \in S)}{\mathbb{P}_{[0,T]}(X_T \in S)}$. For $t \in [0, T]$, set $M_t = \mathbb{E}[Z \mid \mathcal{F}_t]$. Then

$$\mathbb{E}_{\mathbb{P}}[Z \mid \mathcal{F}_t] = \frac{1}{\mathbb{P}_{[0,T]}(X_T \in S)} \mathbb{P}_{[0,T]}(X_T \in S \mid \mathcal{F}_t) = \frac{1}{\mathbb{P}_{[0,T]}(X_T \in S)} h(t, X_t),$$

using the Markov property. We show that M_t is a martingale. First, we observe that $\mathbb{E}[|M_t|] \le \frac{1}{\mathbb{P}_{[0,T]}(X_T \in S)} < \infty$. Secondly, we see that $\mathbb{E}[M_t \mid \mathcal{F}_s] = \mathbb{E}[\mathbb{E}[Z \mid \mathcal{F}_t] \mid \mathcal{F}_s] = \mathbb{E}[Z \mid \mathcal{F}_s] = M_s$ by the tower property of filtrations. Hence, it is a martingale. Moreover, since (M_t) is a uniformly integrable martingale (bounded by $\frac{1}{\mathbb{P}_{[0,T]}(X_T \in S)}$), (Revuz & Yor, 1999, Ch. II, Thm. 2.3) we have

$$M_t \to M_T = Z \in L^1 \quad \text{and a.s., as } t \to T. \tag{41}$$

For each $\varepsilon > 0$, define a probability measure on $\mathcal{F}_{T-\varepsilon}$ by

$$\mathbb{Q}^{\varepsilon}(A) := \mathbb{E}_{\mathbb{P}}[\mathbf{1}_A M_{T-\varepsilon}], \qquad A \in \mathcal{F}_{T-\varepsilon}.$$

If $0 < \varepsilon' < \varepsilon$, then $\mathbb{Q}^{\varepsilon}$ and $\mathbb{Q}^{\varepsilon'}$ agree on $\mathcal{F}_{T-\varepsilon}$:

$$\mathbb{Q}^{\varepsilon'}(A) = \mathbb{E}_{\mathbb{P}}[\mathbf{1}_A M_{T-\varepsilon'}] = \mathbb{E}_{\mathbb{P}}\big[\mathbf{1}_A \, \mathbb{E}_{\mathbb{P}}[M_{T-\varepsilon'} \mid \mathcal{F}_{T-\varepsilon}]\big] = \mathbb{Q}^{\varepsilon}(A), \tag{42}$$

since $\{M_t\}_{t\ge 0}$ is a martingale. By Carathéodory's extension theorem (Ash & Doléans-Dade, 2000, Thm. 1.3.10) increasing sigma-fields, there is a unique probability measure $\mathbb{Q}$ on $\mathcal{F}_{T-} := \sigma\big(\bigcup_{t<T} \mathcal{F}_t\big)$ such that $\mathbb{Q}|_{\mathcal{F}_{T-\varepsilon}} = \mathbb{Q}^{\varepsilon}$ for all $\varepsilon > 0$.

Let $A \in \mathcal{F}_{T-}$; by taking a sequence $A_n \in \mathcal{F}_{t_n}$ with $t_n \to T$ such that $\mathbf{1}_{A_n} \to \mathbf{1}_A$ in L^1 (Ash & Doléans-Dade, 2000, Thm. 1.3.11). By passing to a subsequence on both sides, we upgrade to a.s. pointwise convergence.

$$\mathbb{Q}(A_n) = \mathbb{E}_{\mathbb{P}}[\mathbf{1}_{A_n} M_{t_n}] = \frac{\mathbb{P}_{[0,T]}(A_n \cap \{X_T \in S\})}{\mathbb{P}_{[0,T]}(X_T \in S)} \tag{43}$$

$$\implies \int \mathbf{1}_{A_n} d\mathbb{Q} = \frac{1}{\mathbb{P}_{[0,T]}(X_T \in S)} \int \mathbf{1}_{A_n} M_{t_n} d\mathbb{P} \tag{44}$$

Let $n \to \infty$ and use dominated convergence on both sides (everything bounded by 1) to obtain that

$$\implies \int \mathbf{1}_A d\mathbb{Q} = \frac{1}{\mathbb{P}_{[0,T]}(X_T \in S)} \int \mathbf{1}_A Z d\mathbb{P} = \frac{\mathbb{P}_{[0,T]}(A \cap \{X_T \in S\})}{\mathbb{P}_{[0,T]}(X_T \in S)} \tag{45}$$

$$\implies \mathbb{Q}(A) = \mathbb{P}_{[0,T]}(A \mid X_T \in S). \tag{46}$$

Define the conditional process X^S as the coordinate process under $\mathbb{Q}$ on Ω with coordinate map $X_t(\omega) = \omega(t)$, and set

$$X_t^S := X_t, \quad \text{as a random variable on } (\Omega, \mathcal{F}_{T-}, \mathbb{Q}), \quad 0 \le t < T. \tag{47}$$

This follows from the regularity assumptions imposed on the process.

Now, assumption-1 gives us that X_t is a feller process. Therefore, from (Revuz & Yor, 1999, Ch. III, Thm. (2.7)), it admits a cadlag modification such that the modification has a pathwise left-limit everywhere. Let us call the cadlag modification of X_t^S as $X_t^{S'}$. Thus, $X_{T-}^S := \lim_{t \to T} X_t^{S'}$ everywhere. Since a feller process is also stochastically continuous $\mathbb{P}(X_{T-}^S \neq X_T^S) = 0$. Hence $X_T^S = \lim_{t \to T} X_t^{S'} \mathbb{P}$-a.s. Further, since $X_t^{S'} = X_t^S \mathbb{P}$-a.s., we have that $X_T^S = \lim_{t \to T} X_t^S \mathbb{P}$-a.s. Since $\mathbb{Q}$ is absolutley continuous with respect to $\mathbb{P}$, these statements hold $\mathbb{Q}$-a.s. as well. Moreover, for any Borel $A \subset \mathscr{E}$, the event $\{X_{T-}^S \in A\}$ belongs to $\mathcal{F}_{T-}$ because $X_{T-}^S := \lim_{n \to \infty} X_{T-1/n}^{\prime S} \forall \omega \in \Omega$ where $X_t^{\prime S}$ is the cadlag modification of X_t^S. Note that each $X_t^{\prime S}$ is $\mathcal{F}_{T-}$ measurable and the pointwise limit of measurable functions is also measurable. Hence, applying Eq. 46 with A replaced by $\{X_{T-}^S \in A\}$ yields

$$\mathbb{Q}(X_{T-}^S \in A) = \mathbb{P}_{[0,T]}(X_T \in A \mid X_T \in S), \qquad A \in \mathcal{B}(\mathscr{E}).$$

Further, since $\mathbb{Q}(X_{T-}^S \in A) = \mathbb{Q}(X_T^S \in A)$, we get that

$$\mathbb{Q}(X_T^S \in A) = \mathbb{P}_{[0,T]}(X_T \in A \mid X_T \in S), \qquad A \in \mathcal{B}(\mathscr{E}).$$

This completes the proof.

■

To convert this into a feller process, we observe that the zeroes of h could create issues. To mitigate this, we define our new process on the same path space, but now modify our observable space. For each $s \in [0, T]$, define $U_s := \{x : h(s, x) > 0\}$. Clearly

$$p^S(s, t, x, A) = \int_A p(s, t, x, dy) \frac{h(t, y)}{h(s, x)} \quad \text{for } x \in U_s$$

Claim E.2. *$p^S(s, t, x, A)$ is a valid markov transition function on U_s.*

We firstly observe that for each $t \in [0, T]$, the set U_t is measurable since it is the preimage of the measurable set $\mathbb{R}^+$ under the random variable $h(t, X_t)$

Proof. 1. $p^S(s, t, x, A) = \frac{1}{h(s,x)} \int_A p(s, t, x, dy) h(t, y)$. Since $p(s, t, x, dy)$ is measurable, and $h(s, x) > 0$ on U_s, therefore $x \mapsto p^S(s, t, x, A)$ is also measurable on U_s.

2. We observe that

$$\int_{U_t} p^S(s,t,x,dy) = \int_{U_t} p(s,t,x,dy)\frac{h(t,y)}{h(s,x)} = \frac{1}{h(s,x)}\int_E p(s,t,x,dy)h(t,y)$$

The second equality follows from the fact that $h(t,y) = 0 \,\forall y \in E\backslash U_t$, Now, by *Chapman Kolmogorov*, we know that

$$p(s,T,x,S) = \int_E p(s,t,x,dy)p(t,T,y,S)$$

Now simply by definition, we know that $h(s,x) = p(s,T,x,S)$ and $h(t,y) = p(t,T,y,S)$. Thus,

$$\frac{1}{h(s,x)}\int_E p(s,t,x,dy)h(t,y) = 1$$

and so $p^S(s,t,x,.)$ defines a probability on U_t.

3. We observe that,

$$\int_{U_t} p^S(s,u,x,dy)p^S(u,t,y,A) = \int_{U_t} p(s,u,x,dy)\frac{h(u,y)}{h(s,x)}\int_A p(u,t,y,dz)\frac{h(t,z)}{h(u,y)} \tag{48}$$

Now, we see that if $h(u,y) = 0$, then $\int_A p(u,t,y,dz)h(t,z) \leq \int_E p(u,t,y,dz)p(t,T,z,S) = p(u,T,y,S) = h(t,y) = 0$. Hence, we have that

$$\int_{U_t} p(s,u,x,dy)\frac{h(u,y)}{h(s,x)}\int_A p(u,t,y,dz)\frac{h(t,z)}{h(u,y)}$$

$$= \frac{1}{h(s,x)}\int_{U_t} p(s,u,x,dy)\int_A p(u,t,y,dz)h(t,z) \tag{49}$$

$$= \frac{1}{h(s,x)}\int_E p(s,u,x,dy)\int_A p(u,t,y,dz)h(t,z)$$

By fubini, we have in 49 that,

$$\frac{1}{h(s,x)}\int_E p(s,u,x,dy)\int_A p(u,t,y,dz)h(t,z) \tag{50}$$

$$= \frac{1}{h(s,x)}\int_A h(t,z)\int_E p(s,u,x,dy)p(u,t,y,dz)$$

The inner integral reduces due to *Chapman Kolmogorov*, and so we get that

$$\frac{1}{h(s,x)}\int_A h(t,z)\int_E p(s,u,x,dy)p(u,t,y,dz) \tag{51}$$

$$= \frac{1}{h(s,x)}\int_A h(t,z)p(s,t,x,dz) = p^S(s,t,x,A)$$

■

We first define the changed initial law p_0^S under the new measure $\mathbb{Q}$ as

$$p_0^S(A) = \int_A \frac{h(0,y)p_0(dy)}{\mathbb{P}(X_T \in S)} \tag{52}$$

Claim E.3. *The process $X^S = (\Omega, (X_t^S)_{t\in[0,T]}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{Q})$ is markov with transition function p^S and initial law $p_0^S(.)$*

Proof. 1. We first observe that the initial law of X_0^S under $\mathbb{Q}$ is given as $\mathbb{Q}(X_0^S \in A) = \mathbb{P}(X_0 \in A \mid X_T \in S)$ from 46. By Bayes', we have that $\mathbb{P}(X_0 \in A \mid X_T \in S) = \int_A p_0(dy \mid X_T \in S) = \int_A \frac{h(0,y)p_0(dy)}{\mathbb{P}(X_T\in S)} = p_0^S(A)$

2. Given an arbitrary bounded measurable $f : E \to \mathbb{R}$, we have that

$$\mathbb{E}_{\mathbb{Q}}[f(X_t^S) \mid \mathcal{F}_s] = \mathbb{E}_{\mathbb{Q}}[f(X_t) \mid \mathcal{F}_s] \tag{53}$$

From Baldi (2017)[Chapter 4, Theorem 4.1], we have that the conditional expectation Z of a random variable X_t with respect to a sigma-algebra $\mathcal{F}_s$ is defined as a $\mathcal{F}_s$-measurable random variable such that for all $B \in \mathcal{F}_s$, we have

$$\int_B X_t dP = \int_B Z dP \tag{54}$$

The existence and uniqueness upto a measure zero set is given by the Radon Nikodym Theorem [Chp 1] Baldi (2017). Let us define $Z' \coloneqq \mathbb{E}_{\mathbb{Q}}[f(X_t) \mid \mathcal{F}_s]$. Thus, for all $B \in \mathcal{F}_s$, we have that

$$\int_B Z' d\mathbb{Q} = \int_B f(X_t) d\mathbb{Q} \tag{55}$$

From 43, we know for a $\mathcal{F}_k$-measurable function h, we have that $\int_E h d\mathbb{Q} = \int_E h M_k d\mathbb{P}$ where $M_k = \mathbb{E}_{\mathbb{P}}[Z \mid \mathcal{F}_k]$. Further, we know that Z' is $\mathcal{F}_s$-measurable and $f(X_t)$ is $\mathcal{F}_t$-measurable. So, we have that

$$\int_B Z' M_s d\mathbb{P} = \int_B Z' d\mathbb{Q} = \int_B f(X_t) d\mathbb{Q} = \int_B f(X_t) M_t d\mathbb{P} \tag{56}$$

Let us define by $Z \coloneqq E_{\mathbb{P}}[f(X_t)M_t \mid \mathcal{F}_s]$. Then we know that

$$\int_B Z' M_s d\mathbb{P} = \int_B Z d\mathbb{P} \tag{57}$$

By the uniqueness of the conditional expectation, we obtain that $Z' M_s = Z$ $\mathbb{P}$ a.s. Thus, we have that

$$Z' h(s, X_s) = \mathbb{E}_{\mathbb{P}}[f(X_t)M_t \mid \mathcal{F}_s] = \int_E f(y) h(t, y) p(s, t, X_s, dy) \quad \mathbb{P} \text{ a.s. and so } \mathbb{Q} \text{ a.s.} \tag{58}$$

Consider the set $A = \{\omega : M_s(\omega) = h(s, X_s(\omega)) = 0\}$. Then $\mathbb{Q}(A) = \int_A M_s(\omega) d\mathbb{P}(w) = 0$. Thus consider, the new r.v. Z'' which is defined as

$$Z'' \coloneqq \begin{cases} \frac{1}{h(s,X_s))} \int_E f(y) h(t, y) p(s, t, X_s, dy) & \text{if } h(s, X_s) > 0 \\ f(X_s) & \text{otherwise} \end{cases} \tag{59}$$

We observe that $Z'' = \int_E f(y) p^S(s, t, X_s, dy)$ $\mathbb{Q}$ a.s. and since $h(s, X_s) > 0$ $\mathbb{Q}$ a.s., we have that $Z' = Z''$ $\mathbb{Q}$ a.s. and so

$$\mathbb{E}_{\mathbb{Q}}[f(X_t^S) \mid \mathcal{F}_s] = \mathbb{E}_{\mathbb{Q}}[f(X_t) \mid \mathcal{F}_s] = \int_E f(y) p^S(s, t, X_s, dy) \quad \mathbb{Q}\, a.s. \tag{60}$$

Thus, the new process X^S is indeed markov with transition function p^S. ■

E.2 Proof of Theorem 4.2

Now that we have defined the transition function p^S, we take care of the zeros of h, formally we do this by considering the time homogenized version of X_t^S on an extended observable space which is a subset of $E \times \mathbb{R}^+$

Claim E.4. *Consider the r.v. $\tau = \inf\{t \geq 0 : h(t, X_t) = 0\}$. Then τ is a valid stopping time.*

Proof. The proof is straightforward from Baldi (2017). If h is continuous, then the set $h(t, X_t) > 0$, is an open subset of $E \times [0, T]$. From Baldi (2017)[Proposition 3.7], the statement follows. ■

Next, we show that the process so defined X_t^S is indeed a feller process. To do this, let us first define the function p^S, taking values from and to U as (with some abuse in notation) Let us first define a canonical point ∂_h separate from E. Let us also define $U' \subset E \times [0, T]$ as $U' = \{(s, x) : h(s, x) > 0\}$ and $U'' = \{(t, \partial_h) : t \in [0, T]\}$. Let $\mathcal{B}(E \times [0, T])$, be the standard borel algebra. Endow U' with the subspace topology from $E \times [0, T]$ and U'' with the product topology on $[0, T]$ and $\{\partial_h\}$

(since ∂_h is a singleton, it is the trivial topology). Let us define $U \coloneqq U' \sqcup U''$ and the sigma-algebra $\mathcal{U} \coloneqq \sigma(\{A \cap U' : A \in \mathcal{B}\} \sqcup U'')$. Thus, $(U, \mathcal{U})$ is a valid observable space. We also redefine X_t^S as

$$X_t^S(\omega) = \begin{cases} (\omega(t), t) & \text{if } t < \tau(\omega) \\ (\partial_h, t) & \text{if } t \geq \tau(\omega) \end{cases}$$

We can now define a transition function on U given as $r(h, (s,x), \bar{A} = (t, A))$ given by

$$r(t', (s,x), \bar{A} = (t, A)) = \begin{cases} \begin{cases} p^S(s,t,x, A - \partial_h) & \text{if } x \neq \partial_h \\ 1_A(x) & \text{if } x = \partial_h \end{cases} & \text{if } t = s + t' \\ 0 & \text{if } t \neq s + t' \end{cases}$$

Claim E.5. *r as defined above is well-defined and a valid transition function on U.*

Proof. We observe that it is well defined since if $x \neq \partial_h$, then $h(s,x) > 0$, and so $p^S(s,t,x,dy)$ is indeed well-defined there. The proof of being a valid transition function follows from the time homogenization argument of B.3. ■

Remark E.6. In general, a stopped markov process is *submarkovian* i.e., $\int_E p(s,t,x,dy) \leq 1$ however under the measure $\mathbb{Q}$, we have that $\{h(s,x) > 0\}$ $\mathbb{Q}$ a.s., hence $\mathbb{Q}(\tau \leq t) = 0$

Claim E.7. *X_t^S defined above is a time-homogenous markov process with observable space $(U, \mathcal{U})$, transition function r, and initial law on U given as $p_0^S(dt, dy) = p_0^S(dy)1_{t=0}(dt)$. Further, this process is feller on U.*

Proof. Given a function $f \in \mathcal{C}_0(U)$, and the time homogeneous markov semigroup T_h, we would like to show that $T_h f \in \mathcal{C}_0(U)$. Please note that any symbol h present in the subscript of T_h or present as a sum $s + h$ is supposed to represent time. Usage is clear from context. First, we have that

$$T_h f(s,x) = \int_U f(t,y) r(h, (s,x), (dt, dy))$$

We will work this out in two cases: **Case 1:** $x \neq \partial_h$, then $T_h f(s,x) = \int_{U_{s+h}} f(s+h, y) p^S(s, s+h, x, dy) = \frac{1}{h(s,x)} \int_{U_{s+h}} f(s+h,y) p(s, s+h, x, dy) h(s+h, y)$. Now, we know that $h(s,x)$ is continuous in a neighbourhood of (s,x). We also know that since $h(s+h, y)$ is continuous in (s,y) on U' (since it is positive there). Let us define the extension of $h(t,y)$ to U'' as $h^{U''}(t,y) = 0$. Now, we have $h^{U''}(t,y)$ defined on the entirety of U. Since U' and U'' share no common open sets, $h^{U''}$ is continuous on U. Since it is bounded above by 1, for any $f(t,y) \in \mathcal{C}_0(U)$ we have that $f(t,y)h(t,y) \in \mathcal{C}_0(U)$. Now consider the continuous extension (Assumption 2 of B.4) of $f(x,t)h(x,t)$ onto E. Now, we know that $T_h^{orig} f(s,x)h(s,x) = \int_E f(s+h,y)h(s+h,y)p(s, s+h, x, dy) = \int_{U_{s+h}} f(s+h,y)h(s+h,y)p(s, s+h, x, dy)$. By the original feller hypothesis, since $fh \in \mathcal{C}_0(E \times [0,T])$ we obtain that $T_h^{orig} fh$ is continuous on $E \times [0,T]$, and therefore its restriction onto U' is also continuous. Thus, $\int_{U_{s+h}} f(s+h,y)p(s,s+h,x,dy)h(s+h,y)$ is in $\mathcal{C}_0(U)$. We also know that $h(s,x) = \int_{U_t} p(s,t,x,dy)h(t,y)$. Hence

$$\frac{1}{h(s,x)} \int_{U_{s+h}} f(s+h,y)p(s,s+h,x,dy)h(s+h,y)$$

$$= \frac{\int_{U_{s+h}} f(s+h,y)p(s,s+h,x,dy)h(s+h,y)}{\int_{U_{s+h}} p(s,s+h,x,dy)h(s+h,y)}$$

Given that $f(s+h,y) < \epsilon$ outside some compact interval K_ϵ, we know that we can choose an x with large enough magnitude that

$$\frac{\int_{K_\epsilon} p(s,s+h,x,dy)h(s+h,y)}{\int_{U_{s+h}} p(s,s+h,x,dy)h(s+h,y)} < \epsilon / B$$

where $B = \sup_{x \in K_\epsilon} |f(s+h,x)|$ (by Assumption 3 of B.4). Then

$$\frac{\int_{U_{s+h}} f(s+h,y)p(s,s+h,x,dy)h(s+h,y)}{\int_{U_{s+h}} p(s,s+h,x,dy)h(s+h,y)} = \frac{\int_{K_\epsilon} f(s+h,y)p(s,s+h,x,dy)h(s+h,y)}{\int_{U_{s+h}} p(s,s+h,x,dy)h(s+h,y)}$$

$$+\frac{\int_{U_{s+h}\setminus K_\epsilon} f(s+h,y)p(s,s+h,x,dy)h(s+h,y)}{\int_{U_{s+h}} p(s,s+h,x,dy)h(s+h,y)} \leq B.\epsilon/B + \epsilon \leq 2\epsilon$$

Thus, $T_h f(s,x)$ decays to 0. By previous arguments, it is continuous as well, and so it is in $\mathcal{C}_0(U)$.
Case 2: $x = \partial_h$, in which case $T_h f(s,x) = \int_U f(t,y) r(h,(s,x),(dt,dy)) = f(s+h,x) \in \mathcal{C}_0(U)$ ■

We would now like to show that $\lim_{h\to 0} \|T_h f - f\|_\infty = 0$.

Proof. **Case 1:** We show this statement on U'. Firstly, we observe that $T_h f - f = \frac{1}{h}(T_h^{orig} hf - hf)$ for a continuous extension of hf from U' to $E \times [0,T]$ as in the previous argument. Consider a compact interval K_ϵ such that $|T_h f| < \epsilon/2$ and $|f| < \epsilon/2$ for all $(s,x) \in U' \setminus K_\epsilon$. On K_ϵ, define c_ϵ as $c_\epsilon \coloneqq \inf_{(s,x)\in K_\epsilon} h(s,x)$. This is bounded away from 0 since $h(s,x)$ is positive on U' and K_ϵ is compact. Now, choose h small enough such that $|T_h^{orig} hf - hf| < c_\epsilon \epsilon \, \forall (s,x) \in K_\epsilon$. This is possible due to the uniform bound T_h^{orig}. Then, $|T_h f - f| = |\frac{1}{h}(T_h^{orig} hf - hf)| \leq c_\epsilon \epsilon / c_\epsilon = \epsilon \, \forall (s,x) \in K_\epsilon$. For all $(s,x) \in U' \setminus K_\epsilon$, we know that $|T_h f - f| \leq |T_h f| + |f| \leq \epsilon/2 + \epsilon/2 = \epsilon$. Thus, we have that $|T_h f - f| < \epsilon \, \forall (s,x) \in U'$.

Case 2: For all $(s,x) \in U''$, $T_h f - f = f_h - f$ where $f_h(s,y) = f(s+h,y)$. Since $[0,T]$, is already compact, $f(.,y)$ upgrades to being uniformly continuous on $[0,T]$. Therefore, its convergence is uniform.
Hence, the new process is feller on U. ■

We set up some extra notation here, whenever we refer to a function with a subscript of time, say t, it is the restriction of that function onto its space coordinate. That is, $f^t : E \to \mathbb{R}, x \mapsto f(t,x)$
Theorem E.8. *For all $f \in \mathcal{T} = \mathcal{C}_C^\infty(U')$, the time dependent generator $\mathcal{L}_t^S$ of X_t^S satisfies the following equation*

$$\mathcal{L}_t^S f^t = \frac{\mathcal{L}_t h^t f^t - f^t \mathcal{L} h^t}{h^t}$$

where $f^t(x) : U_t \to \mathbb{R}$, $f^t(x) = f(t,x)$.

Proof. Recall that the time conditional generator is defined as

$$\mathcal{L}_s^S f^s(x) \coloneqq \lim_{\delta \downarrow 0} \frac{\mathbb{E}[f(s+\delta, X_{s+\delta}) - f(s+\delta, X_s) \mid X_s = x]}{\delta}$$

We show that this limit exists in supremum norm for all $f \in \mathcal{T}$. First we observe that the time dependent generator is 0 on $x = \partial_h$ since $X_{s+\delta} = X_s$ if $X_s = \partial_h$. Therefore, we restrict ourselves to U'. Let us define $g(s,\delta,x)$

$$g(s,\delta,x) \coloneqq \frac{\int_{U_{s+\delta}} f(s+\delta,y)p(s,s+\delta,x,dy)h(s+\delta,y)}{h(s,x)} - f(s+\delta,x)$$

Then, first we see that $\lim_{\delta\downarrow 0} g(s,\delta,x)/\delta = \mathcal{L}_s^S f^s(x)$. Now, we have that

$$g(s,\delta,x) = \frac{\int_{U_{s+\delta}} f(s+\delta,y)p(s,s+\delta,x,dy)h(s+\delta,y)}{h(s,x)} - \frac{f(s+\delta,x)h(s+\delta,x)}{h(s,x)} \tag{61}$$

$$+\frac{f(s+\delta,x)h(s+\delta,x)}{h(s,x)} - \frac{f(s+\delta,x)h(s,x)}{h(s,x)}$$

Secondly, we also have that $h(s,x) = \int_{U_{s+\delta}} h(s+\delta,y)p(s,s+\delta,x,dy)$. From Assumption 6 of B.4, $s \mapsto \mathcal{L}_s f^s$ is continuous. We further assume that $h^s f^s \in \mathcal{D}(\mathcal{L}_s)$ [1]. Through standard properties of derivatives, we know that $\int_{U_{s+\delta}} f(s+\delta,y)h(s+\delta,y)p(s,s+\delta,x,dy) - f(s+\delta,x)h(s+\delta,x) = \delta\mathcal{L}_s f^s h^s(x) + o(\delta,x)$, where $o(\delta,x)$ is a superlinear function of δ and tends to 0 as $\delta \to 0$. Similarly, $\int_{U_{s+\delta}} h(s+\delta,y)p(s,s+\delta,x,dy) - h(s+\delta,x) = \delta\mathcal{L}_s h^s(x) + o'(\delta,x)$. Also since, the original convergence was uniform, o, o' are just functions of δ. Rewriting 61, we get that

$$g(s,\delta,x) = \frac{\delta\mathcal{L}_s(h^s f^s)(x) + o(\delta)}{h(s,x)} - \frac{\delta f(s+\delta,x)\mathcal{L}_s h^s(x) + o'(\delta)}{h(s,x)}$$

Then,

$$\begin{aligned} &\frac{g(s,\delta,x)}{\delta} - \frac{\mathcal{L}_s(h^s f^s)(x)}{h(s,x)} + \frac{f(s,x)\mathcal{L}_s h^s(x)}{h(s,x)} = \\ &\frac{o'(\delta) - o(\delta)}{h(s,x)\delta} + \frac{f(s,x)\mathcal{L}_s h^s(x)}{h(s,x)} - \frac{f(s+\delta,x)\mathcal{L}_s h^s(x)}{h(s,x)} \end{aligned} \tag{62}$$

$$\implies \lim_{\delta\downarrow 0}\left|\frac{g(s,\delta,x)}{\delta} - \frac{\mathcal{L}_s(h^s f^s)(x)}{h(s,x)} + \frac{f(s,x)\mathcal{L}_s h^s(x)}{h(s,x)}\right| = 0$$

Now fix an $\epsilon > 0$. Let f be 0 outside some compact set $K \subset B_R$, then choose δ small enough such that the RHS of 62 is smaller than ϵ for all $|x| < 2R$. This can be done since convergence in a compact interval upgrades to uniform convergence. For all $|x| > 2R$, observe that in 61, only the first term survives outside the ball of $2R$, further there is a small δ' such that

$$g(s,\delta',x)/\delta' \le \|f\|_\infty \frac{\int_{B_R} p(s,s+\delta',x,dy)h(s+\delta',y)}{\delta' h(s,x)} < \epsilon$$

from Assumption 4 of B.4. Clearly as $\delta \to 0$, $g(s,\delta,x) \to 0$ for $|x| > B_{2R}$. Then choose $\delta'' = \min(\delta,\delta')$ to obtain $|g(s,\delta'',x) - \lim g(s,\delta'',x)| < \epsilon$ uniformly. ■

Therefore, the generator is defined for all $f \in C_c^\infty(U')$. Further, Assumptions 1,5,6,7 of Appendix B.4 still hold. The expected number of jumps in any time interval $[s,t]$ is still finite since the new path measure $\mathbb{Q}$ is absolutely continuous with respect to $\mathbb{P}$. We have showed that the new action of the generator holds for all $f \in C_c^\infty(U')$. Further $t \mapsto \mathcal{L}_t^S f$ is continuous since $t \mapsto \mathcal{L}_t(h^t f)$ and $t \mapsto \mathcal{L}_t(h^t)$ are both continuous, and h^t is continuous where $h(t,x) > 0$. Further, $t \mapsto \langle p_t^S, f\rangle = \langle p_t, fh^t\rangle/\mathbb{P}(X_1 \in S)$ which is continuous. Hence, one can now apply Theorem 3.1.

E.3 Proof of Corollary 4.3

We show this for all 3 components separately. The key theorem we will be working is

$$\mathcal{L}_t^S f(x) \;=\; \frac{\mathcal{L}_t\big(h(t,\cdot)f\big)(x) - f(x)\,\mathcal{L}_t h(t,\cdot)(x)}{h(t,x)}.$$

E.3.1 Flows

Assuming that h is differentiable, we get that

$$\mathcal{L}_t^S f(x) = \frac{1}{h(t,x)}\left[\nabla_x(f(x)h(t,x))^T u_t(x) - f(x)\nabla_x(h(t,x)^T u_t(x)\right] \tag{63}$$

By product rule, we get that

$$\begin{aligned} \mathcal{L}_t^S f(x) = \frac{1}{h(t,x)}[&h(t,x)\nabla_x(f(x))^T u_t(x) + f(x)\nabla_x(h(t,x))^T u_t(x) \\ &- f(x)\nabla_x(h(t,x))^T u_t(x)] \\ = \nabla_x(f(x))^T u_t(x) \end{aligned}$$

Hence, $u_t^S(x) = u_t(x)$

[1] Note that in general this might mean assuming h^s is smooth, however once invoked, it can be shown that the generator also assumes such a form under much weaker conditions such as $\mathcal{C}_2(U')$

E.3.2 DIFFUSIONS

Here, we assume that h is twice differentiable. Then, we get that

$$\mathcal{L}_t^S f(x) = \frac{1}{2h(t,x)} \left[\nabla_x^2 (f(x)h(t,x))^T \cdot \sigma_t^2(x) - f(x)\nabla_x^2 h(t,x) \cdot \sigma_t^2(x) \right] \tag{64}$$

The first term can be expanded by the product rule as

$$\begin{aligned} \nabla_x^2 (f(x)h(t,x)) &= \nabla_x \left(h(t,x)\nabla_x f(x) + f(x)\nabla_x h(t,x) \right) && (65) \\ &= h(t,x)\nabla_x^2 f(x) + \nabla_x f(x)\nabla_x h(t,x)^T && (66) \\ &+ \nabla_x h(t,x)\nabla_x f(x)^T + f(x)\nabla_x^2 h(t,x) && (67) \end{aligned}$$

Then, we have that

$$\mathcal{L}_t^S f(x) = \frac{1}{2}\nabla_x^2 f(x) \cdot \sigma_t^2(x) + \frac{1}{2h(t,x)} \left[\nabla_x f(x)\nabla_x h(t,x)^T + \nabla_x h(t,x)\nabla_x f(x)^T \right] \cdot \sigma_t^2(x) \tag{68}$$

The frobenius inner product between two matrices A and uv^T is $\mathrm{tr}(A^T(uv^T)) = \mathrm{tr}(v^T A^T u)$ (the cyclic property holds for vectors as well). Applying this to 68, we get that,

$$\mathcal{L}_t^S f(x) = \frac{1}{2}\nabla_x^2 f(x) \cdot \sigma_t^2(x) + \nabla_x f(x)^T \left(\sigma_t^2(x) \frac{\nabla_x h(t,x)}{h(t,x)} \right)$$

Hence, there is additional drift of $\sigma_t^2(x)\frac{\nabla_x h(t,x)}{h(t,x)} = \sigma_t^2(x)\nabla_x \log h(t,x)$

E.3.3 JUMPS

For jumps, we know that $\mathcal{L}_t f(x) = \int_E Q_t(dy,x)[f(y) - f(x)]$. Expanding we get that,

$$\begin{aligned} \mathcal{L}_t^S f(x) &= \frac{1}{h(t,x)} \int_E Q_t(dy,x)[f(y)h(t,y) - f(x)h(t,x)] && (69) \\ &- \frac{f(x)}{h(t,x)} \int_E Q_t(dy,x)[h(t,y) - h(t,x)] && (70) \\ &= \int_E Q_t(dy,x)\frac{h(t,y)}{h(t,x)}[f(y) - f(x)] - \cancel{\int_E Q_t(dy,x)f(x)} + \cancel{\int_E Q_t(dy,x)f(x)} && (71) \end{aligned}$$

Hence, we obtain that

$$Q_t^S(dy,x) = Q_t(dy,x)\frac{h(t,y)}{h(t,x)}$$

Since $h(t,y) \leq 1$ and $h(t,x) > 0$, the measure is still finite.

Combining all the three, we get the result for Corollary 4.3.

E.4 PROOF OF COROLLARY 4.4

We first observe that the conditioned generator of each column satisfies

$$\mathcal{L}_t^{S(j)} f(x) = \frac{\mathcal{L}_t^{(j)}(fh^t)(x) - f(x)\mathcal{L}_t^{(j)}(h^t)(x)}{h(t,x)}$$

This is precisely the doob's transform applied to the generator of each column individually (keeping all the other columns fixed). Moreover, we also know that

$$\mathcal{L}_t f(x) = \sum_j \mathcal{L}_t^{(j)} f(x)$$

Now,

$$\mathcal{L}_t^S f(x) = \frac{\mathcal{L}_t(fh^t)(x) - f(x)\mathcal{L}_t(h^t)(x)}{h(t,x)} \tag{72}$$

$$= \frac{\sum_j \mathcal{L}_t^{(j)}(fh^t)(x) - \sum_j f(x)\mathcal{L}_t^{(j)}(h^t)(x)}{h(t,x)} \tag{73}$$

$$= \sum_j \mathcal{L}_t^{S(j)} f(x) \tag{74}$$

Hence, the conditioned process can be simulated by simulating each of the conditioned generators column by column, with the input taken from the whole row.

E.5 Proof of Proposition 4.5

We first observe that by definition

$$p_{1|t}(dz \mid x) \propto p_{\text{data}}(dz) p_t(x \mid z)$$

At $t = 0$, the per-sample path forgets its initial starting point. This is reflected in the fact that $p_0(x \mid z) = p_0(x) = p_{\text{noise}}$. From this, we obtain that $p_{1|t}(dz \mid x) \propto p_{\text{data}}(dz)$. Now if $p(0, 1, x, dz) = p_{1|0}(dz \mid x)$, then $h(0, x) = p(0, 1, x, S) = p_{1|0}(S \mid x) = p_{\text{data}}(S) = \mathbb{P}(X_1 \in S)$. In this case,

$$p_0^S(dy) = p_0(dy)\frac{h(0,y)}{\mathbb{P}(X_1 \in S)} = p_0(dy)\frac{\mathbb{P}(X_1 \in S)}{\mathbb{P}(X_1 \in S)} = p_0(dy)$$

Now consider the process with X_t^z with generator $\mathcal{L}_t^z$. Since this is the bridge of the markov process X_t, the machinery of the doob's transform applies here as well. Given a point $(x, t) \in E \times [0, 1]$, the transition function $p(t, 1, x, .)$ is a measure on E which is crucially *absolutely continuous* with respect to p_{data} since $p(1, t, x, .) = p_{1|t}(. \mid x)$ and $p_{1|t}(\cdot \mid x) \propto p_t(x \mid z)p_{\text{data}}(\cdot)$, where the analytic paths $p_t(x \mid z$ are usually smooth. Therefore, from the Radon Nikodym Theorem [Chp 1] Baldi (2017), it admits a density $h_z(t, x)$ with respect to p_{data}. The generator of the X_t^z is then given by $\mathcal{L}_t^z f = [\mathcal{L}_t(h_z f) - f\mathcal{L}_t(h_z)]/h_z$. Now, by construction, the generator $\widetilde{\mathcal{L}}_t^S$ is defined as

$$\widetilde{\mathcal{L}}_t^S f(x) = \mathbb{E}_{z \sim p_{1|t}^S(\cdot|x)}\left[\mathcal{L}_t^z f(x)\right]$$

where

$$p_{1|t}^S(\cdot \mid x) = \frac{p_{1|t}(\cdot \mid x)\mathbf{1}_{z \in S}}{\int_S p_{1|t}(dz \mid x)}$$

Under the assumption that $p_{1|t}(dz \mid x) = p(t, 1, x, dz)$, we get that

$$\widetilde{\mathcal{L}}_t^S f(x) = \mathbb{E}_{z \sim p_{1|t}^S(\cdot|x)}\left[\mathcal{L}_t^z f(x)\right] \tag{75}$$

$$= \int_E \mathcal{L}_t^z f(x) p_{1|t}^S(dz \mid x) \tag{76}$$

$$= \int_E \mathcal{L}_t^z f(x) \frac{p_{1|t}(dz \mid x)\mathbf{1}_{z \in S}}{\int_S p_{1|t}(dz \mid x)} \tag{77}$$

$$= \int_E \left[\frac{\mathcal{L}_t(h_z f) - f\mathcal{L}_t(h_z)}{h_z}\right] \frac{p_{1|t}(dz \mid x)\mathbf{1}_{z \in S}}{\int_S p_{1|t}(dz \mid x)} \tag{78}$$

$$\tag{79}$$

Now, $\int_S p_{1|t}(dz \mid x) = \int_S p(t, 1, x, dz) = p(t, 1, x, S) = h(t, x)$. Hence,

$$\widetilde{\mathcal{L}}_t^S f(x) = \frac{1}{h(t,x)} \int_S \left[\mathcal{L}_t(h_z f) - f\mathcal{L}_t(h_z)\right] p_{\text{data}}(dz) \tag{80}$$

where we used that $h_z(t, x)p_{\text{data}}(dz) = p(t, 1, x, dz) = p_{1|t}(dz \mid x)$. Since the generator $\mathcal{L}_t$ is linear in its arguments, we can push the integral inside the generator [2] to get that

$$\widetilde{\mathcal{L}}_t^S f(x) = \frac{1}{h(t,x)}\left[\mathcal{L}_t\left(f(x)\int_S h_z(t,x)p_{\text{data}}(dz)\right) - f(x)\mathcal{L}_t\left(\int_S h_z(t,x)p_{\text{data}}(dz)\right)\right] \tag{81}$$

[2] we assume some form of uniform continuity in this case

Since again we have that $h_z(t,x)p_{\text{data}}(dz) = p(t,1,x,dz)$, then

$$\int_S h_z(t,x)p_{\text{data}}(dz) = \int_S p(t,1,x,dz) = p(t,1,x,S) = h(t,x)$$

Hence, Equation 80 reduces to

$$\widetilde{\mathcal{L}}_t^S f(x) = \frac{1}{h(t,x)}\left[\mathcal{L}_t(hf) - f\mathcal{L}_t(h)\right] = \mathcal{L}_t^S f(x)$$

Hence, we get that $\widetilde{\mathcal{L}}_t^S f(x) = \mathcal{L}_t^S f(x)$.

E.6 Proofs for training matching

We want to show that

$$\widetilde{\mathcal{L}}_t^S f(x) = \mathbb{E}_{z\sim p^S_{1|t}(.|x)}[\mathcal{L}_t^z f(x)] \tag{82}$$

Proof. Let us denote by $k_{t+h|t}(.\mid x,z)$ to be the transition kernel for the conditional markov process X_t^z with conditional generator $\mathcal{L}_t^z$ which solves the KFE for the marginals given by $p_t(.\mid z)$. Let $p_t^S(.)$ denote the time marginals of the conditioned process. Then, we have that

$$\partial_t\langle p_t^S, f\rangle = \lim_{h\to 0}\frac{1}{h}\langle p_{t+h}^S, f\rangle - \langle p_t^S, f\rangle \tag{83}$$

$$= \lim_{h\to 0}\frac{1}{h}[\mathbb{E}_{x'\sim p^S_{t+h}(.)}[f(x')] - \mathbb{E}_{x\sim p_t^S(.)}[f(x)]] \tag{84}$$

$$= \lim_{h\to 0}\frac{1}{h}[\mathbb{E}_{z\sim p^S_{\text{data}}(\cdot),\, x'\sim p_{t+h}(.|z)}[f(x')] - \mathbb{E}_{z\sim p^S_{\text{data}}(\cdot),\, x\sim p_t(.|z)}[f(x)]] \tag{85}$$

$$= \lim_{h\to 0}\frac{1}{h}[\mathbb{E}_{z\sim p^S_{\text{data}}(\cdot),\, x\sim p_t(.|z),\, x'\sim k_{t+h|t}(.|x,z)}[f(x') - f(x)]] \tag{86}$$

$$= \lim_{h\to 0}\frac{1}{h}\mathbb{E}_{x\sim p_t^S(\cdot)}\left[\mathbb{E}_{z\sim p^S_{1|t}(.|x)}\left[\mathbb{E}_{x'\sim k_{t+h|t}(.|x,z)}[f(x')] - f(x)\right]\right] \tag{87}$$

$$= \mathbb{E}_{x\sim p_t(.|S)}\left[\mathbb{E}_{z\sim p^S_{1|t}(.|x)}\underbrace{\left[\lim_{h\to 0}\frac{1}{h}\left[\mathbb{E}_{x'\sim k_{t+h|t}(.|x,z)}[f(x')] - f(x)\right]\right]}_{:=\mathcal{L}_t^z f(x)}\right] \tag{88}$$

$$\tag{89}$$

Hence, we get that

$$\partial_t\langle p_t^S, f\rangle = \mathbb{E}_{x\sim p_t^S(\cdot)}\left[\mathbb{E}_{z\sim p^S_{1|t}(.|x)}[\mathcal{L}_t^z f(x)]\right] \tag{90}$$

Under the assumptions of B.4, the KFE determines the marginals and so the generator defined as

$$\widetilde{\mathcal{L}}_t^S f(x) = \mathbb{E}_{z\sim p^S_{1|t}(.|x)}[\mathcal{L}_t^z f(x)] \tag{91}$$

determines the marginals $p_t^S(\cdot)$. Here, we used that the convergence is uniform in x. ■

Secondly, we show that this holds for any linear parametrization of $\widetilde{\mathcal{L}}_t^S$ as well. A linear parametrization of $\widetilde{\mathcal{L}}_t^S$ is defined through a convex subset of an inner product space W, linear operator $\mathcal{K} : \mathcal{C}_c^\infty(E) \to W$ and a parametrization $F_t : E \to W$, such that $\mathcal{L}_t f(x) = \langle \mathcal{K}f(x), F_t(x)\rangle_W$. The proof simply relies on the linearity of the inner product in the second argument. Define

$$\mathcal{L}_t^z f(x) = \langle \mathcal{K}f(x), F_t^z(x)\rangle_W$$

Then,

$$\widetilde{\mathcal{L}}_t^S f(x) = \mathbb{E}_{z\sim p^S_{1|t}(.|x)}[\langle \mathcal{K}f(x), F_t^z(x)\rangle_W] \tag{92}$$

$$= \left\langle \mathcal{K}f(x), \underbrace{\mathbb{E}_{z\sim p^S_{1|t}(.|x)}[F_t^z(x)]}_{:=\widetilde{F}_t^S(x)}\right\rangle \tag{93}$$

$$\implies \widetilde{\mathcal{L}}_t^S f(x) = \langle \mathcal{K}f(x), \widetilde{F}_t^S(x)\rangle_W \tag{94}$$

E.7 Proof of Consistency

We adapt the proof from (Holderrieth et al., 2025, Section C.3.2). One can express a Bregman Divergence from a convex potential $\phi : \Omega \to \mathbb{R}$ as

$$D(a, b) = \phi(a) - [\phi(b) + \langle a - b, \nabla\phi(b)\rangle]$$

Following Holderrieth et al. (2025), for presentation purposes, we write $D(a, b) = A(a) + \langle a, B(b)\rangle + C(b)$ where $A(a) = \phi(a)$, $B(b) = \nabla\phi(b)$, $C(b) = -\phi(b) + \langle b, \nabla\phi(b)\rangle$.

$$L^S_{gm}(\theta) = \mathbb{E}_{t\sim Unif, x\sim p^S_t(\cdot)}[D(\widetilde{F}^S_t(x), F^{S,\theta}_t(x))] \tag{95}$$

$$= \mathbb{E}_{t\sim Unif, x\sim p^S_t(\cdot)}[A(\widetilde{F}^S_t(x)) + \left\langle \widetilde{F}^S_t(x), B(F^{S,\theta}_t(x))\right\rangle + C(F^{S,\theta}_t(x))] \tag{96}$$

$$= \mathbb{E}_{t\sim Unif, x\sim p^S_t(\cdot)}[A(\widetilde{F}^S_t(x)) + \left\langle \int_E F^z_t(x) p^S_{1|t}(dz \mid x), B(F^{S,\theta}_t(x))\right\rangle + C(F^{S,\theta}_t(x))] \tag{97}$$

$$= \mathbb{E}_{t\sim Unif, x\sim p^S_t(\cdot)}[A(\widetilde{F}^S_t(x)) + \int_E \left\langle F^z_t(x), B(F^{S,\theta}_t(x))\right\rangle p^S_{1|t}(dz \mid x) + C(F^{S,\theta}_t(x))] \tag{98}$$

where the last equality follows from the fact that inner product is linear in the first argument.[3]. We also observe that $A(\widetilde{F}^S_t(x))$ is independent of θ, and therefore can be replaced with $\int_E A(F^z_t(x)) p^S_{1|t}(dz \mid x)$ in the expectation. The last term is independent of z anyways and can be pulled inside the integral to yield

$$\mathbb{E}_{t\sim Unif, x\sim p^S_t(\cdot)}\left[\int_E \left[A(F^z_t(x)) + \left\langle F^z_t(x), B(F^{S,\theta}_t(x))\right\rangle + C(F^{S,\theta}_t(x))\right] p^S_{1|t}(dz \mid x)\right]$$

$$= \mathbb{E}_{t\sim Unif, x\sim p^S_t(\cdot), z\sim p^S_{1|t}(\cdot|x)}[D(F^z_t(x), F^{S,\theta}_t(x))]$$

$$= \mathbb{E}_{t\sim Unif, z\sim p^S_{\text{data}}, x\sim p_t(\cdot|z)}[D(F^z_t(x), F^{S,\theta}_t(x))] = L^S_{cgm}(\theta)$$

which is additive to $L^S_{gm}(\theta)$ upto constants in θ.

E.8 Necessity of Assumption 3 of B.4

Consider a jump process running from $t = 0$ to $t = 1$ on observable space $\mathbb{R}$ with a constant in time jump rate $\lambda_t(x) = \frac{1}{1+x}$ to only $y = 0$ at any point $x \in \mathbb{R}$. Such a process is feller, and satisfies the assumptions of B.4 except Assumption 3. Here, the set $\{y = 0\}$ is of positive measure at $t = 1$. Then, the transition function of this process is given by

$$p(s, t, x, dy) = \begin{cases} 1 - e^{-\lambda(x)(t-s)} & \text{if } y = 0, \\ e^{-\lambda(x)(t-s)} & \text{if } y = x, \end{cases}$$

Further $p(s, t, 0, dy) = 1$ if $y = 0$ and 0 everywhere else. Then, we get that

$$h(t, x) = p(t, 1, x, 0) = \mathbb{P}(X_1 = 0 \mid X_t = x) = 1 - e^{-\lambda(x)(T-t)}$$

Then, let us consider the new transition function $p(s, t, x, dy)\frac{h(t,y)}{h(s,x)}$, which is non-zero if $y = 0$ or $y = x$, and so the action of the transition operator $T_{s,t}$ on some $f \in \mathcal{C}_0(\mathbb{R})$ is given as

$$T_{s,t}f(x) = f(0)\frac{1 - e^{-\lambda(x)(t-s)}}{1 - e^{-\lambda(x)(T-s)}} + f(x)e^{-\lambda(x)(t-s)}\frac{1 - e^{-\lambda(x)(T-t)}}{1 - e^{-\lambda(x)(T-s)}}$$

Now as $|x| \to \infty$, the first term tends to $\frac{t-s}{T-s} \gg 0$, and so $T_{s,t}f(x)$ does not decay to 0. This means that the conditioned process is not feller anymore. This gives that in general, unless Assumption 3 imposed, the resulting process may not be feller.

[3] In general infinite integrals may not commute with inner products, but here assuming finiteness and boundedness of the integral, we could express the Lebesgue integral as a sequence of approximations across partitions, and then apply dominated convergence to push the inner product inside the integral

Proof. This can be generalised, and given a process that violates Assumption 3, there exists a compact set $K \subset U_t$ such that

$$\left| \frac{\int_K p(s,t,x,dy)h(t,y)}{\int_{U_t} p(s,t,x,dy)h(t,y)} \right| \gg 0 \text{ as } |x| \to \infty \quad \text{for some } s \leq t$$

Now consider some $f \geq 0 \in \mathcal{C}_c^\infty(U_t)$ which is lower bounded away from 0 on K. Then

$$|T_{s,t}f(x)| = \left| \frac{\int_K p(s,t,x,dy)h(t,y)f(t,y)}{\int_{U_t} p(s,t,x,dy)h(t,y)} \right| + \left| \frac{\int_{U_t \setminus K} p(s,t,x,dy)h(t,y)f(t,y)}{\int_{U_t} p(s,t,x,dy)h(t,y)} \right| \tag{99}$$

$$\geq \left| \frac{\int_K p(s,t,x,dy)h(t,y)f(t,y)}{\int_{U_t} p(s,t,x,dy)h(t,y)} \right| \tag{100}$$

$$\geq (\inf_{x \in K} f(x)) \left| \frac{\int_K p(s,t,x,dy)h(t,y)}{\int_{U_t} p(s,t,x,dy)h(t,y)} \right| \tag{101}$$

$$\gg 0 \tag{102}$$

since $f \geq 0$ everywhere and bounded away from zero on K. Hence, the conditioned process is not feller. This completes the proof. ■

F Explicit examples of conditioning

F.1 Closed-form conditioner for TabbyFlow

TabbyFlow predicts a factorized endpoint Gaussian distribution with mean $\mu_\theta(x,t)$ (§3). For unconditional generation, this is indeed unbiased, since minimizing the squared loss against the endpoint z does recover the posterior mean. For a numerical column j, this distribution is $\mathcal{N}(\mu_j, \varrho_t^2)$, where μ_j is the j-th component of $\mu_\theta(x,t)$. Its variance follows the fixed schedule

$$\varrho_t^2 = 1 - (1 - \sigma_{\min})\, t^2, \tag{103}$$

which is shared across numerical columns and is not learned. The same $\sigma_{\min}$ enters the path, $x_t = (1 - (1 - \sigma_{\min})t)\,\epsilon + t\,z$, and the velocity, $u_\theta = (\mu_\theta - (1 - \sigma_{\min})x)/(1 - (1 - \sigma_{\min})t)$, which §3 writes with $\sigma_{\min} = 0$. In the reference implementation of TabbyFlow, $\sigma_{\min}$ is hard-coded to 0.01, and we use the same value. For a categorical column j, the endpoint distribution is categorical with probabilities μ_j. Let $S = \prod_j S_j$, with $S_j \subseteq E_j$, and let Φ and φ denote the standard normal CDF and density. Since both the posterior and S factorize across columns, and so conditioning can be performed independently for each column. For a numerical column with $S_j = [a,b]$, define $\alpha = (a - \mu_j)/\varrho_t$ and $\beta = (b - \mu_j)/\varrho_t$. The conditional mean is then given as

$$\mu_j + \varrho_t \frac{\varphi(\alpha) - \varphi(\beta)}{\Phi(\beta) - \Phi(\alpha)}, \tag{104}$$

the mean of $\mathcal{N}(\mu_j, \varrho_t^2)$ truncated to $[a,b]$. One-sided constraints follow by taking $a = -\infty$ or $b = \infty$. When $\Phi(\beta) - \Phi(\alpha)$ underflows, which can occur near $t = 1$ as ϱ_t approaches $\sqrt{\sigma_{\min}}$, we use the endpoint of $[a,b]$ closest to μ_j which is obtained as the limiting value of Eq. 104. For a categorical column with $S_j = A \subseteq E_j$, the conditional mean of the one-hot endpoint is obtained by setting the entries of μ_j outside A to zero and renormalizing the remainder. Equivalently, the corresponding logits are set to $-\infty$. Unconstrained columns keep their original mean μ_j. The conditioned velocity is obtained by replacing $\mu_\theta(x,t)$ with these conditional means in the TabbyFlow velocity, after which sampling uses the original solver unchanged. As ϱ_t decreases toward $\sqrt{\sigma_{\min}}$, the truncated-Gaussian correction approaches projection of μ_j onto $[a,b]$. It is, however a biased estimator, since the posterior need not be Gaussian. In the unconditional case, minimizing squared error indeed recovers the posterior mean, with the Gaussian assumption never being used anywhere. In the unconditional case this does not however matter, since the mean is correctly parameterized.

G UNCONDITIONAL MODELS: DESCRIPTION AND DETAILS

This section describes all unconditional models used in the paper. §G.1 fixes notation, preprocessing, and the time convention. §G.2 states the settings of the published baselines and the quantities of theirs that later sections use. §G.3 describes the symmetric core that replaces their flattened MLP, and §G.4 presents TABJUMP. Finally, §G.5 places all models in the framework of Theorem 3.1 and gives the full version of Table 1.

G.1 NOTATION, PREPROCESSING, AND TIME

Notation. A row has n columns, of which m are numerical and $n - m$ categorical. As in §2, E_j denotes the state space of column j, and $E = \mathbb{R}^m \times F$ with $F = \prod_{j \text{ categorical}} E_j$. We write x_j for column j, $x = (x^{\text{num}}, x^{\text{cat}})$ for a row split by type, z for a clean row, and x_t for a noisy row at time t. Masked models augment each categorical state space with a mask state M, giving $\bar{E}_j = E_j \cup \{\mathsf{M}\}$ and $\bar{F} = \prod_{j \text{ categorical}} \bar{E}_j$ as in §3. In a one-hot encoding of $\bar{E}_j$, M is the last of $|E_j| + 1$ entries. We write $x^{j \leftarrow v}$ for x with column j set to v, and $\varsigma_j(t)$ for the noise level of numerical column j at time t in a diffusion model. Column-specific quantities are indexed by j. The index is omitted when clear from context, for example in g_t. Each network predicts $\mu_\theta(x, t)_j$ per column (§3.2). This quantity is a denoised value for numerical columns of diffusions, endpoint probabilities for categorical columns, bin probabilities for TABJUMP, and an endpoint mean for TABBYFLOW.

Preprocessing. All neural models use the preprocessing of Zhang et al. (2024), adopted unchanged by TABDIFF and TABBYFLOW. Each numerical column is mapped by a quantile transform to a standard normal marginal. The transform is fitted on the training rows only, using the number of training rows divided by 30 quantiles and clipping this number to $[10, 1000]$; missing values are set to zero before fitting. Each categorical column is encoded by integers following the sorted order of its string values, with M assigned code $|E_j|$. The prediction target is treated as an ordinary column of its corresponding type. Generated rows are mapped back through the inverse numerical transform and categorical code table. TABPC uses its own preprocessing (§G.2).

Time. Throughout the paper, t runs from noise at $t = 0$ to data at $t = 1$ (§2), following generator matching and TABBYFLOW. The diffusion models TABDIFF and TABSYN use the opposite convention, with data at their time 0. A schedule defined by these papers at time τ is therefore evaluated here at $\tau = 1 - t$. In particular, their noise levels decrease with our t, and the diffusion coefficient g of the noising SDE appears in Table 1 as $g_t = g(1 - t)$, consistent with §3. We retain the original names of model-specific schedules and write their time arguments explicitly when needed.

G.2 BASELINES AND SETTINGS

For each baseline, we use the default settings from the paper and the reference implementation. The symmetric variants keep these settings and replace only the network core, as described in §G.3.

TABDIFF. Numerical columns follow variance-exploding paths $x_{t,j} = z_j + \varsigma_j(t)\epsilon_j$, with $\epsilon_j \sim \mathcal{N}(0, 1)$ and learned, decreasing noise levels $\varsigma_j(t)$. Categorical columns follow masked chains, with learned unmasking probabilities $\alpha_j(t)$ and rates $\lambda_j(t) = \alpha_j'(t)/(1 - \alpha_j(t))$. A masked column jumps to category y at rate $\lambda_j(t)p_{1|t}(z_j = y \mid x)$.

TABBYFLOW. TABBYFLOW applies variational flow matching (Eijkelboom et al., 2024) to all columns, using one-hot categorical encodings. The network predicts factorized Gaussian and categorical endpoint distributions, whose means $\mu_\theta(x, t)$ determine the velocity (§3). The fixed endpoint variance and the sampling velocity are regularized with $\sigma_{\min} = 0.01$ (§F.1).

TABSYN and TABSYNFLOW. Both models use the VAE of Zhang et al. (2024), selected on the validation split. It encodes each column as $\xi_j \in \mathbb{R}^{d_\xi}$ with $d_\xi = 4$. We denote the flattened latent row by $\xi \in \mathbb{R}^{nd_\xi}$, reserving z for clean endpoints. TABSYN applies variance-exploding diffusion with a shared noise level $\varsigma(t)$. TABSYNFLOW trains a flow along the linear path from Gaussian noise to ξ and predicts the velocity v_θ directly.

TABPC. TABPC (Scassola et al., 2026) is a probabilistic circuit. Permuting the columns permutes its circuit in the Chow–Liu construction, so it needs no symmetric variant.

G.3 THE SYMMETRIC CORE: ARCHITECTURE AND COLUMN SYMMETRY

G.3.1 ARCHITECTURE AND HYPERPARAMETERS

We describe the symmetric core used in all experiments. The same module is used by TABDIFF-*sym*, TABBYFLOW-*sym*, and TABJUMP through their respective wrappers. We use the notation of §3.2 and §G.1: n columns, representation width d, and L blocks indexed by ℓ.

Tokenizer. For a numerical column j, we use the periodic embedding of Gorishniy et al. (2022),

$$r_j = \mathrm{ReLU}\Big(W_j\big[\sin(2\pi\zeta_j x_j);\ \cos(2\pi\zeta_j x_j)\big] + b_j\Big), \qquad \zeta_j \in \mathbb{R}^K,\ W_j \in \mathbb{R}^{d\times 2K},\ b_j \in \mathbb{R}^d, \tag{105}$$

where the K frequencies ζ_j are learned separately for each column and initialized as $\zeta_j \sim \mathcal{N}(0, \sigma_f^2 I)$, with σ_f the initialization scale. For a categorical column, the wrapper represents x_j as a one-hot vector over its state space, $\bar{E}_j$ for TABDIFF-*sym* and TABJUMP and E_j for TABBYFLOW-*sym*, and we set

$$r_j = x_j^\top C_j + c_j, \qquad C_j \in \mathbb{R}^{|\bar{E}_j|\times d},\ c_j \in \mathbb{R}^d \tag{106}$$

with $\bar{E}_j$ read as E_j when there is no mask state. These are the per-column feature tokenizers of the FT-Transformer (Gorishniy et al., 2021). Its class token is created and discarded and therefore plays no role.

Core. The core follows the recursion in Eq. 6. ATTN_{θ_0} is multi-head self-attention with H heads of width d/H. The query, key, value, and output maps are all $\mathbb{R}^d \to \mathbb{R}^d$ and are applied row-wise, with the attention softmax taken over the n columns. We use neither positional encodings nor attention masks. The feed-forward network is

$$\mathrm{FFN}_{\theta_0} = \mathrm{Linear}(d, \gamma d) \to \mathrm{GELU} \to \mathrm{Linear}(\gamma d, d), \tag{107}$$

applied independently to each row. The time embedding is computed once per forward pass as

$$e(t) = \mathrm{Linear}(d,d) \circ \mathrm{SiLU} \circ \mathrm{Linear}(d,d)\big(\textsc{Sinusoid}(t)\big) \in \mathbb{R}^d, \tag{108}$$

where SINUSOID is the sinusoidal embedding of Karras et al. (2022). Each block applies its own $\mathrm{SiLU} \to \mathrm{Linear}(d,d)$ projection to $e(t)$ before adding it to every column representation. The main text absorbs this block-specific projection into e_t. A final layer normalization gives R'.

Detokenizer. For a numerical column,

$$\phi_j = w_j'^\top r_j', \qquad w_j' \in \mathbb{R}^d. \tag{109}$$

For a categorical column,

$$\phi_j = W_j' r_j' + b_j', \qquad W_j' \in \mathbb{R}^{|E_j|\times d}, \tag{110}$$

which gives logits over the same state space. Both detokenizers are specific to column j. In the notation of §G.1, ϕ_j denotes the raw network output that is mapped to $\mu_\theta(x,t)_j$ by the model-specific wrapper. For numerical columns, this mapping is the EDM preconditioning of Karras et al. (2022) for TABDIFF-*sym* and the identity for TABBYFLOW-*sym*. For categorical columns, it is a softmax, with the M logit removed where present.

Hyperparameters. All experiments use

$$d = 128, \qquad L = 6, \qquad H = 8, \qquad \gamma = 4, \qquad K = 48, \qquad \sigma_f = 0.05,$$

with no dropout. For a numerical column, the column-specific parameters are $\theta_j = (\zeta_j, W_j, b_j, w_j')$; for a categorical column, they are $\theta_j = (C_j, c_j, W_j', b_j')$. The column-specific noise schedule is also included in θ_j when required by the wrapper. All remaining parameters are shared and belong to θ_0.

Table 3: Parameter counts of Φ_θ **in millions**. The symmetric core replaces the flattened MLP of Eq. 5 with the self-attention blocks of Eq. 6, reducing the parameter count by 5.3–$7.4\times$. TABJUMP is built directly on the symmetric core and has no original counterpart.

Dataset	TABDIFF	TABDIFF-*sym*	TABBYFLOW	TABBYFLOW-*sym*	TABJUMP
SHOPPERS	21.29	2.94	21.29	2.93	3.54
MAGIC	21.18	2.90	21.18	2.90	3.97
BEIJING	21.20	2.86	21.20	2.86	3.05
DEFAULT	21.39	3.05	21.39	3.04	4.35
ADULT	21.25	2.86	21.25	2.85	3.04
NEWS	21.79	3.82	21.79	3.82	7.02
DIABETES	21.65	4.12	21.65	4.11	4.06
FB-COMMENTS	21.85	3.65	21.85	3.65	6.08

Wrappers. For TABDIFF-*sym*, the symmetric core replaces the original core of Eq. 5, together with its tokenizer and detokenizer, inside the preconditioning and mixed-type diffusion of TABDIFF. Schedules, loss, training, and sampling are as in §G.2. For TABBYFLOW-*sym*, an adapter converts the flat state into $(x^{\text{num}}, x^{\text{cat}})$ before the core and concatenates the outputs before returning them to the flow-matching wrapper of Guzmán-Cordero et al. (2025). TABJUMP uses the same core with every column treated as categorical (§G.4).

Parameter counts. Table 3 compares the original networks with their symmetric variants. Removing the dense layers on the flattened grid reduces the parameter count by 5.3–$7.4\times$, with a mean reduction of $6.7\times$, for both TABDIFF and TABBYFLOW. Their original networks have nearly identical parameter counts because both use the same original network.

G.3.2 COLUMN SYMMETRY: DEFINITION AND PROOFS

Let $\mathcal{P}_n = \left\{P \in \{0,1\}^{n\times n} : \sum_i P_{ij} = \sum_j P_{ij} = 1\right\}$ be the set of $n \times n$ permutation matrices. Each $P \in \mathcal{P}_n$ represents a permutation π of $[n]$ through $P_{ij} = \mathbf{1}[i = \pi(j)]$. We write P_π for the matrix of π. The permutation moves column j to position $\pi(j)$,

$$(\pi x)_{\pi(j)} = x_j, \tag{111}$$

and acts on the representation grid by permuting its rows,

$$\pi R = P_\pi R. \tag{112}$$

Column-specific parameters move with their columns, while the shared parameters remain fixed:

$$\pi\theta = (\theta_0, \theta_{\pi^{-1}(1)}, \ldots, \theta_{\pi^{-1}(n)}). \tag{113}$$

Column types, state spaces, and noise schedules are part of θ_j and are permuted with the corresponding column.

Definition G.1 (Column equivariance). A network Φ is *column equivariant* if, for every permutation π, parameter set θ, state $x \in E$, and $t \in [0,1]$,

$$\Phi_{\pi\theta}(\pi x, t) = \pi\, \Phi_\theta(x, t). \tag{114}$$

Since the tokenizer and detokenizer act independently on each column, column equivariance reduces to a property of the core.

Lemma G.2 (Per-column modules). *For every π, θ, x,*

$$\textsc{Tokenizer}_{\pi\theta}(\pi x) = \pi\, \textsc{Tokenizer}_\theta(x), \tag{115}$$

and, for every representation grid R',

$$\textsc{Detokenizer}_{\pi\theta}(\pi R') = \pi\, \textsc{Detokenizer}_\theta(R'). \tag{116}$$

Proof. Row $\pi(j)$ of $\textsc{Tokenizer}_{\pi\theta}(\pi x)$ is computed from $(\pi x)_{\pi(j)} = x_j$ using parameters $(\pi\theta)_{\pi(j)} = \theta_j$. It therefore equals $\textsc{Tokenizer}_{\theta_j}(x_j)$, which is row j of $\textsc{Tokenizer}_\theta(x)$. The same argument applies to the detokenizer. ■

Lemma G.3 (Set encoder). *Every block of Eq. 6 satisfies*

$$\textsc{Block}_{\theta_0}(P_\pi R, t) = P_\pi \,\textsc{Block}_{\theta_0}(R, t), \tag{117}$$

and hence so does the symmetric core.

Proof. All row-wise operations commute with P_π, including LN, FFN_{θ_0}, the query, key, value, and output maps, and the residual additions. The time embedding is added as $\mathbf{1}_n e_t^\top$, and $P_\pi \mathbf{1}_n = \mathbf{1}_n$, so this term is invariant. For one attention head, let

$$Q = RW_Q, \qquad K = RW_K, \qquad V = RW_V.$$

Replacing R by $P_\pi R$ gives $P_\pi Q$, $P_\pi K$, and $P_\pi V$. Hence

$$\mathrm{softmax}\Big(\frac{P_\pi Q K^\top P_\pi^\top}{\sqrt{d/H}}\Big) = P_\pi \,\mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d/H}}\Big) P_\pi^\top, \tag{118}$$

because row-wise softmax commutes with simultaneous permutations of rows and columns. Multiplying by $P_\pi V$ gives P_π times the original head output. Concatenating heads and applying the output map preserves this property. The feed-forward block, residual connections, and layer normalizations are row-wise, so composing the sublayers and inducting over ℓ proves the result for the full core. ■

Proposition G.4 (The symmetric core is column equivariant). *The network of §3.2 using the symmetric core Eq. 6 satisfies Eq. 114.*

Proof. By Lemma G.2, $\textsc{Tokenizer}_{\pi\theta}(\pi x) = P_\pi R$. By Lemma G.3, $\textsc{Core}_{\theta_0}(P_\pi R, t) = P_\pi R'$, where θ_0 is unchanged by π. Applying Lemma G.2 again gives $\textsc{Detokenizer}_{\pi\theta}(P_\pi R') = \pi\,\Phi_\theta(x, t)$, which proves Eq. 114. ■

Proposition G.5 (The flattened MLP does not enforce column equivariance). *Let the core contain*

$$G_t = \mathrm{vec}^{-1} \circ \mathrm{MLP}_{\theta_0}(\cdot, t) \circ \mathrm{vec},$$

as in Eq. 5. For a column permutation π, let $P = P_\pi \otimes I_d$. Column equivariance requires

$$\mathrm{MLP}_{\theta_0}(Pv, t) = P\,\mathrm{MLP}_{\theta_0}(v, t) \qquad \textit{for all } v, t. \tag{119}$$

An unconstrained dense MLP is not constrained to satisfy Eq. 119. Thus, unlike the symmetric core, the flattened-MLP architecture does not enforce column equivariance.

Proof. Since

$$\mathrm{vec}(P_\pi R) = (P_\pi \otimes I_d)\,\mathrm{vec}(R) = P\,\mathrm{vec}(R),$$

equivariance of G_t is equivalent to Eq. 119. Already for a linear map $v \mapsto Wv + b$ from $\mathbb{R}^{nd}$ to itself, this requires

$$WP = PW, \qquad Pb = b, \tag{120}$$

for every column permutation π. A dense layer places no such constraints on W or b, and therefore admits parameter settings that distinguish fixed column positions. Adding further unconstrained dense layers does not impose these constraints. Hence the flattened-MLP architecture is not column equivariant by construction. The surrounding ENC and DEC blocks are equivariant, but cannot make the overall architecture equivariant for arbitrary parameters of the intervening MLP. ■

Consequences and numerical check. The losses of TABDIFF, TABBYFLOW, and TABJUMP (§G.4) are sums of per-column terms, so column equivariance of Φ makes the training loss invariant under column reordering. Since each sampler updates a column using its corresponding ϕ_j, its transition kernel is likewise equivariant. The generated distribution therefore does not depend on the order in which the columns are presented. We verify Eq. 114 numerically on trained models using random inputs, times, and column permutations. We measure

$$\Delta_\pi = \frac{\|\Phi_{\pi\theta}(\pi x, t) - \pi\,\Phi_\theta(x, t)\|}{\|\Phi_\theta(x, t)\|}. \tag{121}$$

In double precision, the symmetric core gives $\Delta_\pi \approx 5 \times 10^{-16}$ for both numerical and categorical outputs, while the original networks give $\Delta_\pi = O(1)$. The computed losses are likewise invariant to machine precision for the symmetric models but not for the originals.

G.4 TABJUMP

TABJUMP is a pure-jump model over the full row, with coordinate-wise jumps. Numerical columns are first discretized into quantile bins, so every column has a finite state space, and the symmetric core of §G.3 predicts a distribution over its clean value. The model has no flow or diffusion component and uses no preconditioning. §G.5 describes the same model as a jump process on the numerical values themselves, with the within-bin draw as part of the jump, and shows that this produces the same distribution over rows as the bin chain followed by decoding.

Binned state space. Numerical bins are fitted on the training rows after the quantile transform of §G.1, so their boundaries lie on the standard-normal scale. Let B be the bin budget per column and let V_j be the set of distinct training values of numerical column j. If $|V_j| \le B$, every value receives its own bin, and the column is represented exactly. Otherwise, every value whose relative frequency in the training rows is at least ρ_{atom} receives a degenerate *atom* bin $[v, v]$, and the remaining values are split into interval bins $[l, u)$ of equal empirical mass, placed between consecutive atoms, with the remaining budget shared among the gaps *in proportion to their mass*. Adjacent intervals are merged, narrowest first, whenever rounding exceeds the budget. The result is $B_j \le B$ ordered bins, and $\bar{B}_j$ is this set of bins together with M. Atoms matter because tabular numerical columns often concentrate a large share of their mass on a few repeated values, such as zero. A plain quantile grid would pool such a value with its neighbours, and dequantization would then spread an exact value over an interval.

Decoding maps an atom bin to its exact value. An interval bin is decoded by sampling uniformly within the interval on the transformed scale, followed by the inverse quantile transform of §G.1. This avoids placing all generated values at bin centres. The discretization bounds what any model on these bins can achieve.

Masking Process. The source state is the all-mask row. We have independently for each column j,

$$x_{t,j} = \begin{cases} z_j, & \text{with probability } \alpha(t), \\ \mathsf{M}, & \text{otherwise,} \end{cases}$$

with

$$\alpha(t) = t, \qquad \lambda_t = \frac{\alpha'(t)}{1-\alpha(t)} = \frac{1}{1-t}. \tag{122}$$

This is the masked path of TABDIFF's categorical columns with a fixed linear schedule in place of a learned one. Its target process is a CTMC in which a masked column j jumps to y at rate $\lambda_t\, p_{1|t}(z_j = y \mid x_t)$ and a revealed column never changes.

Neural network. TABJUMP uses the symmetric core of §G.3.1. Since numerical columns are discretized, every column enters through the categorical tokenizer. Numerical columns are one-hot encoded over $\bar{B}_j$ and categorical columns over $\bar{E}_j$. The detokenizer returns logits over the corresponding masked state space. The M logit is set to $-\infty$ before the softmax, giving $\mu_\theta(x,t)_j$, the predicted distribution of the clean value of column j. Time enters through the embedding $e(t)$ of §G.3.1.

Loss. Let $t \sim \mathrm{Unif}[0, 1-\varepsilon_{\mathrm{cut}}]$ and sample x_t from the path above. TABJUMP minimizes the generalized KL generator-matching loss of Holderrieth et al. (2025),

$$\mathbb{E}\left[\lambda_t \sum_{j=1}^{n} \mathbf{1}[x_{t,j} = \mathsf{M}]\big(-\log \mu_\theta(x_t,t)_j[z_j]\big)\right]. \tag{123}$$

We cap the rate using $\varepsilon_{\mathrm{cut}} = 10^{-3}$. As shown in §G.5, Eq. 123 is the CGM loss, up to terms independent of θ, for learned rates $Q_t^\theta(y,x)_j = \lambda_t\, \mu_\theta(x,t)_j[y]\, \mathbf{1}[x_j = \mathsf{M}]$.

Training. Training follows TABDIFF-*sym* with the warm-up and gradient clipping used by TABBYFLOW. We use Adam with learning rate 10^{-3}, batch size 4096, a 100-epoch linear warm-up, gradient clipping at norm 1, learning-rate decay by 0.9 after 50 epochs without improvement, EMA

Algorithm 1 TABJUMP training

Require: training rows; bin budget B; atom threshold ρ_{atom}; cut-off ε_{cut}; epochs T
fit the numerical bins on the training rows (§G.4)
initialize the symmetric core Φ_θ and its EMA parameters $\bar{\theta} \leftarrow \theta$
for epoch $= 1, \dots, T$ **do**
  for each mini-batch **do**
    encode numerical columns into bins to obtain the clean finite-state rows z
    draw $t \sim \text{Unif}[0, 1 - \varepsilon_{\text{cut}}]$ independently for each row
    set $x_{t,j} \leftarrow z_j$ with probability $\alpha(t)$ and M otherwise
    obtain $\mu_\theta(x_t, t)$ by masking the M logits and applying softmax
    take a gradient step on Eq. 123 with gradient clipping; update $\bar{\theta}$
  end for
  update the learning rate
  if epoch $\geq$ 4,000 and the EMA training loss is the lowest so far **then**
    save $\bar{\theta}$
  end if
end for
return the saved $\bar{\theta}$ and fitted bins

Algorithm 2 TABJUMP sampling

Require: trained $\Phi_{\bar{\theta}}$; fitted bins; steps N; cut-off ε_{cut}
$x \leftarrow$ the all-mask row; $t_i \leftarrow i(1 - \varepsilon_{\text{cut}})/N$ for $i = 0, \dots, N$
for $i = 0, \dots, N - 1$ **do**
  $t \leftarrow t_i$; $h \leftarrow t_{i+1} - t_i$
  $\mu \leftarrow \mu_{\bar{\theta}}(x, t)$
  draw $\hat{z}_j \sim \mu_j$ for every masked column j
  if $i < N - 1$ **then**
    independently set $x_j \leftarrow \hat{z}_j$ with probability $1 - \exp(-h\lambda_t)$
  else
    set $x_j \leftarrow \hat{z}_j$ for every masked column j
  end if
end for
decode numerical bins and apply the inverse quantile transform; map categorical codes back
return the decoded row

decay 0.997, and 10,000 epochs. The checkpoint is the averaged model with the lowest training loss after epoch 4,000. Warm-up and clipping are particularly useful because λ_t grows rapidly near $t = 1$.

Sampling. Sampling uses $N = 50$ steps on $t_i = i(1 - \varepsilon_{\text{cut}})/N$. At each step, for each masked column j, we draw $\hat{z}_j \sim \mu_\theta(x, t)_j$ and accept it with probability $1 - \exp(-h\lambda_t)$, where $h = t_{i+1} - t_i$. Numerical bins and categorical codes are then decoded as in §G.4 and §G.1.

Hyperparameters. We use $B = 256$ for all datasets except FB-COMMENTS for which we use $B = 512$. Additionally, $\rho_{\text{atom}} = 0.01$, $\alpha(t) = t$, $\varepsilon_{\text{cut}} = 10^{-3}$, and $N = 50$ sampling steps. The symmetric core uses $d = 128$, $L = 6$, $H = 8$, $\gamma = 4$, and $K = 48$. Training follows the procedure above; parameter counts are given in Table 3.

G.5 TABULAR GENERATIVE MODELS AS INSTANCES OF GENERATOR MATCHING

This subsection extends the characterization of §3 to all generator-matching models of §G.2 and §G.4. Table 4 gives the full version of Table 1. For each model, we identify its state space, the active components and parameters of Eq. 4, the quantity predicted by the network, and the relation between the CGM objective of Eq. 3 and the training loss of §G.2 or §G.4.

Factorized per-sample paths. Every model considered here uses per-sample paths that factorize across columns, $p_t(\cdot \mid z) = \prod_j p_t^{(j)}(\cdot \mid z_j)$, so that the per-sample generator is a sum of single-column generators and the marginal generator decomposes as $\mathcal{L}_t = \sum_j \mathcal{L}_t^{(j)}$, where $\mathcal{L}_t^{(j)}$ updates only column j but its parameters may depend on the full row x (Holderrieth et al., 2025, Props. 4 and 5). In Theorem 3.1 this means that σ_t^2 is diagonal and that every jump changes a single column. Writing $x^{j \leftarrow v}$ for the row x with column j set to v, the jump kernel of such a model is the finite measure

$$Q_t(\mathrm{d}y, x) = \sum_j \sum_{v \neq x_j} Q_t(v, x)_j \, \delta_{x^{j \leftarrow v}}(\mathrm{d}y), \tag{124}$$

where the inner sum runs over the states available to column j in the corresponding model and $Q_t(v, x)_j \geq 0$ is the rate at which column j moves to v. When a column takes continuous values and moves by jumps (like the numerical columns of TABJUMP), the inner sum is an integral against a finite measure on $\mathbb{R}$. Theorem 3.1 also permits jumps that change several columns at once, although the models considered here do not use them. Columns therefore evolve separately given the joint state, and cross-column dependence enters through the full row on which the per-column parameters depend. For the latent models, the same statement holds with latent tokens ξ_j in place of table columns.

Table 4 extends Table 1 to all generator-matching models, including their state spaces, schedules, network parameterizations, and per-sample CGM targets. We next derive TABJUMP's numerical jump kernel, which acts directly on continuous values.

G.5.1 TABJUMP AS AN INSTANCE OF GENERATOR MATCHING

Among the published baselines considered here, jumps are used only for categorical columns. TABJUMP extends the same idea to numerical columns. In the framework of Theorem 3.1, it is a pure-jump process. Numerical column j takes values in $\mathbb{R} \sqcup \{\mathsf{M}\}$ on the transformed scale of §G.1, so the full state space is

$$\Big(\prod_{j \text{ numerical}} (\mathbb{R} \sqcup \{\mathsf{M}\}) \Big) \times \bar{F}.$$

Equivalently, representing the mask of each numerical column by a binary flag gives a state space of the form $\mathbb{R}^m \times F'$ in Theorem 3.1, with $F' = \{0, 1\}^m \times \bar{F}$. Categorical columns follow the masked CTMC of the categorical part of TABDIFF, but with the fixed schedule $\alpha(t) = t$ and $\lambda_t = 1/(1-t)$ from Eq. 122. For a numerical column, the bins of §G.4 define a finite collection of fixed probability measures on $\mathbb{R}$. For bin b of column j, let $\nu_{j,b}$ be the decoding distribution of that bin, which would be a point mass for an atom bin and a uniform distribution on the corresponding interval for an interval bin. Let $b_j(v)$ denote the bin assigned to v by the encoder of §G.4, with $b_j(\mathsf{M}) = \mathsf{M}$. In particular, $\nu_{j,b_j(z_j)}$ is the decoding distribution of the bin containing z_j, that is, the distribution of z_j after an encode–decode round trip. The per-sample kernel of a numerical column is

$$Q_t(\mathrm{d}y, x \mid z)_j = \lambda_t \, \mathbf{1}[x_j = \mathsf{M}] \, \nu_{j,b_j(z_j)}(\mathrm{d}y). \tag{125}$$

Thus, a masked value jumps at rate λ_t to a sample from the decoding distribution of the bin containing z_j, while a revealed value remains fixed. Marginalizing this process gives $x_{t,j} = \tilde{z}_j$ with probability $\alpha(t)$ and M otherwise, where $\tilde{z}_j \sim \nu_{j,b_j(z_j)}$. The process therefore ends at an encode–decode version $\tilde{z}$ of the clean row rather than at z itself. It preserves atom values and bin membership exactly, while replacing the distribution within each interval bin by the fixed decoding distribution $\nu_{j,b}$. The marginal kernel of Eq. 124 follows from the posterior average of Eq. 2,

$$Q_t(\mathrm{d}y, x)_j = \lambda_t \, \mathbf{1}[x_j = \mathsf{M}] \sum_b p_{1|t}\big(b_j(z_j) = b \mid x\big) \, \nu_{j,b}(\mathrm{d}y). \tag{126}$$

The generator of column j is therefore

$$\int_{\mathbb{R}} \big(f(x^{j \leftarrow y}) - f(x)\big) \, Q_t(\mathrm{d}y, x)_j.$$

The network reads the bins $b(x)$ and predicts $\mu_\theta(x, t)_j$, a distribution over the bins of column j (§G.4), so the learned kernel is

$$Q_t^\theta(\mathrm{d}y, x)_j = \lambda_t \, \mathbf{1}[x_j = \mathsf{M}] \sum_b \mu_\theta(x, t)_j[b] \, \nu_{j,b}(\mathrm{d}y). \tag{127}$$

		TABDIFF	TABBYFLOW	TABJUMP
Process state space		$\mathbb{R}^m \times \bar{F}$	$\mathbb{R}^{m+\sum_{j\ \text{categorical}} \lvert E_j \rvert}$	$\prod_{j\ \text{numerical}} (\mathbb{R} \sqcup \{\mathsf{M}\}) \times \bar{F}$
Schedule		$\varsigma_j(t)$, $\alpha_j(t)$ (§G.2)	linear path (§G.2)	$\alpha(t) = t$ (§G.4)
Numerical columns	Components (Eq. 4)	flow + diffusion $u_t = g_t^2 \nabla \log p_t$, $\sigma_t^2 = g_t^2$	flow $u_t \in \mathbb{R}^m, \quad \sigma_t^2 = 0$	jump (masked, mixture kernel) target $Q_t(\mathrm{d}y, x)_j$ nonzero only if $x_j = \mathsf{M}$
	Generator	$\nabla f(x)^\top u_t(x)$ $+\frac{1}{2}\nabla^2 f(x) \cdot \sigma_t^2$	$\nabla f(x)^\top u_t(x)$	$\int_{\mathbb{R}} \big(f(y) - f(x)\big)$ $\times\, Q_t(\mathrm{d}y, x)_j$
	Predicted	μ_θ, such that $s_\theta = (\mu_\theta - x)/\varsigma_j^2$	μ_θ, such that $u_\theta = (\mu_\theta - x)/(1-t)$	μ_θ over bins, such that $Q_t^\theta(\mathrm{d}y, x)_j = \lambda_t\, \mathbf{1}[x_j = \mathsf{M}]$ $\times \sum_b \mu_\theta[b]\, \nu_{j,b}(\mathrm{d}y)$
	CGM loss	$\big\lVert s_\theta(x_t, t)$ $-\nabla \log p_t(x_t \mid z) \big\rVert^2$	$\big\lVert u_\theta(x_t, t)$ $-(z - x_t)/(1-t) \big\rVert^2$	generalized KL of the kernels, equal to the categorical entry on bins (Eq. 123)
Categorical columns	Components (Eq. 4)	jump (masked CTMC) target $Q_t(y, x)$ nonzero only if $x_j = \mathsf{M}$	flow on one-hot coordinates $u_t(x) \in \mathbb{R}^{\lvert E_j \rvert}, \quad \sigma_t^2 = 0$	jump (masked CTMC) target $Q_t(y, x)$ nonzero only if $x_j = \mathsf{M}$
	Generator	$\sum_y \big(f(y) - f(x)\big)\, Q_t(y, x)$	$\nabla f(x)^\top u_t(x)$	$\sum_y \big(f(y) - f(x)\big)\, Q_t(y, x)$
	Predicted	μ_θ, giving $Q_t^\theta(y, x)_j$ $= \lambda_j(t)\, \mu_\theta[y]$ $\times\, \mathbf{1}[x_j = \mathsf{M}]$	μ_θ, giving $u_\theta = (\mu_\theta - x)/(1-t)$	μ_θ, giving $Q_t^\theta(y, x)_j = \lambda_t\, \mu_\theta[y]$
	CGM loss	$\sum_{y \neq x_j} \big(Q_t^\theta(y, x)_j$ $-Q_t(y, x \mid z)_j \log Q_t^\theta(y, x)_j\big)$	$-\log \mu_\theta(x_t, t)_j[z_j]$	$\sum_{y \neq x_j} \big(Q_t^\theta(y, x)_j$ $-Q_t(y, x \mid z)_j \log Q_t^\theta(y, x)_j\big)$

(a) Data-space models.

		TABSYN	TABSYNFLOW
Process state space		$\mathbb{R}^{nd_\xi}$	$\mathbb{R}^{nd_\xi}$
Schedule		$\varsigma(t)$, shared by all coordinates (§G.2)	linear OT path
Latent tokens	Components (Eq. 4)	flow + diffusion $u_t(\xi) = g_t^2 \nabla \log p_t(\xi), \quad \sigma_t^2 = g_t^2 I$	flow $u_t(\xi) \in \mathbb{R}^{nd_\xi}, \quad \sigma_t^2 = 0$
	Generator	$\nabla f(\xi)^\top u_t(\xi) \;+\; \frac{1}{2}\nabla^2 f(\xi) \cdot \sigma_t^2$	$\nabla f(\xi)^\top u_t(\xi)$
	Predicted	μ_θ, giving $s_\theta = (\mu_\theta - \xi)/\varsigma^2$	v_θ, the velocity
	CGM loss	$\big\lVert s_\theta(\xi_t, t) - \nabla \log p_t(\xi_t \mid \xi) \big\rVert^2$	$\big\lVert v_\theta(\xi_t, t) - (\xi - \epsilon) \big\rVert^2$
Original categorical columns		decoded from the generated latent; no separate process component	

(b) Latent-space models.

Table 4: **Full version of Table 1.** For each model, we give its state space, schedule, active generator components of Eq. 4, network parameterization, and CGM loss with its per-sample target. Time runs from noise at $t = 0$ to data at $t = 1$. Here θ denotes learned quantities, $x^{j \leftarrow y}$ replaces column j of x by y, and $f(y)$ abbreviates $f(x^{j \leftarrow y})$. For TABJUMP, $b_j(v)$ is the bin containing v, and $\nu_{j,b}$ is the decoding measure of bin b, a point mass for an atom bin and a uniform distribution for an interval bin (§G.4). For TABDIFF and TABSYN, g_t^2 is the negative time derivative of the squared noise level, and $s_\theta = (\mu_\theta - x)/\varsigma_j(t)^2$ is the score implied by the denoiser. The unmasking rates are $\lambda_j(t) = \alpha_j'(t)/(1 - \alpha_j(t))$ for TABDIFF and $\lambda_t = 1/(1-t)$ for TABJUMP. Per-sample targets are $(z - x_t)/\varsigma_j(t)^2$ for numerical diffusion, $(z - x_t)/(1-t)$ for TABBYFLOW, $\xi - \epsilon$ for latent flow, and $\lambda_j(t)\mathbf{1}[y = z_j]\mathbf{1}[x_j = \mathsf{M}]$ for masked categorical chains. See §G.2 and §G.4 for schedules.

The generalized KL divergence is the rate-weighted cross-entropy $-\lambda_t \log \mu_\theta(x, t)_j[b_j(z_j)]$ on masked coordinates. Summing over columns gives Eq. 123, the categorical objective of TABDIFF with bins in place of categories.

Since the learned kernel depends on x only through $b(x)$ and $\nu_{j,b}$ is supported inside bin b, the bin process $b(X_t)$ is a CTMC in which a masked coordinate j jumps to bin b' at rate $\lambda_t\, \mu_\theta(b(x), t)_j[b']$. This is the chain trained and sampled in §G.4. Numerical values are decoded from $\nu_{j,b}$ given the final bins.

Point masses and within-bin variation are handled by the fixed measures $\nu_{j,b}$, while the network models only the distribution over bins. A numerical interval acts on the bins it intersects and a categorical constraint on its allowed values, so both reduce to finite-state constraints (Appendix H). The cost of this method is that variation inside an interval bin is fixed by its decoding distribution.

H AMORTIZED CONDITIONING: MODULE, LOSSES, TRAINING, AND SAMPLING

We describe the conditioning module of §5. §H.1 describes the architecture. We derive the training losses in §H.2 from Eq. 11, and training and sampling procedures in §H.3 and §H.4. Finally, we discuss their cost in §H.5. Throughout, θ denotes the frozen parameters of an unconditional model from Appendix G, and χ the parameters of the conditioning module. A *rectangular* constraint is a product $S = \prod_j S_j$, where $S_j = [a_j, b_j]$ for a constrained numerical column, $S_j \subseteq E_j$ for a constrained categorical column, and $S_j = E_j$ otherwise. We denote by J_S the constrained columns and $|J_S|$ for the arity of S.

H.1 ARCHITECTURE

Additional Gadgets. Recall from Corollary 4.3 that conditioning adds $\sigma_t^2 \nabla_{\text{num}} \log h$ to the drift and multiplies jump kernels by $h(t, x^{j \leftarrow y})/h(t, x)$. Following Guo et al. (2026), we observe that producing a good approximation to h does not necessarily provide a good approximation to $\nabla h/h$. Therefore, we elect to learn these two corrections with separate networks that share the same *trunk* architecture. Both, the constraint encoding mechanisms and the trunk architecture are agnostic to the parameterization one elects to use for their models (score, noise, or mean predictors).

For TABJUMP every column is finite-state and only the categorical network is used.

Symbol	Meaning
$S = \prod_j S_j$, J_S	constraint set and constrained columns
$[a_j, b_j]$	interval of a constrained numerical column
$s_j \in \mathbb{R}^{d'}$, $\bar{s}_j$	constraint encoding and unconstrained-column encoding
d', L', H', γ'	module width, depth, attention heads, and feed-forward factor
d_b, d_a	bound-embedding and active-flag width dimensions
$\Delta_\chi(x, t, S)_j$	correction added to the numerical denoiser
$\log h_\chi(x, t, S)$	log correction function for jump guidance
$\ell_\chi(x, t, S)_j[y]$	TABJUMP network: $\log h$ at the row $x^{j \leftarrow y}$, predicted from the current row x

Table 5: Notation for the conditioning module.

Constraint encoding. Each column receives an encoding determined by S_j. For a constrained numerical column, we represent $S_j = [a_j, b_j]$ by its centre $\frac{1}{2}(a_j + b_j)$ and log-width $\log(b_j - a_j)$. Each scalar is embedded by a two-layer network into $\mathbb{R}^{d_b}$, and the scalars and their embeddings are concatenated. All numerical constraints in the paper have finite endpoints (§I.3). Their bounds are mapped to the preprocessed scale using the quantile transform of §G.1 before being passed to the module. For a constrained categorical column j, we sum the frozen category embeddings of the allowed values, $\sum_{k \in S_j} \text{ReLU}(C_{j,k})$, where $C_{j,k}$ is the embedding of category k of column j in the base tokenizer. This representation is *invariant* to the ordering of the values in S_j and uses the same category representation as the base model. Further, if $S_j^{(1)} \subset S_j^{(2)}$ then $\sum_{k \in S_j^{(1)}} \text{ReLU}(C_{j,k}) \leq \sum_{k \in S_j^{(2)}} \text{ReLU}(C_{j,k})$ pointwise. Then, the numerical and categorical representations are projected to a common width d', giving $s_j \in \mathbb{R}^{d'}$. An unconstrained column receives a fixed per-column

learned vector $\bar{s}_j \in \mathbb{R}^{d'}$. We implement this as the follows

$$s_j^{\text{final}} = s_j \mathbf{1}[j \in J_S] + \bar{s}_j(1 - \mathbf{1}[j \in J_S])$$

We additionally concatenate an embedding of width d_a of the indicator $\mathbf{1}[j \in J_S]$. If the column is unconstrained, we instead concatenate with the embedding of $1 - \mathbf{1}[j \in J_S]$. For TABJUMP, every column has a finite alphabet, so both numerical and categorical constraints are represented as allowed sets and bound embeddings are ignored. Categorical constraints are encoded as above. A numerical interval $[a_j, b_j]$ is mapped to the bins of column j that intersect the interval. If the bins containing a_j or b_j extend beyond the interval, they are still included in full. Their decoding is described in §H.4.

Trunk. The trunk follows the symmetric core of §G.3.1 at reduced width. The frozen tokenizer of the base model maps x_t to column tokens $r_j \in \mathbb{R}^d$. For TABDIFF-*sym*, numerical inputs use the same input scaling as the base model. Each token is projected to width d', combined with the time embedding $e(t)$, and passed through a linear layer with SiLU activation. This is done since the input at time t is the noisy input x_t, while the constraint inputs represent clean $t = 1$ values. The result is concatenated with s_j and the active-flag embedding and mapped back to $\mathbb{R}^{d'}$, giving r'_j. The grid $R' = [r'_1; \ldots; r'_n]$ is processed by L' blocks of Eq. 6 with H' heads and feed-forward factor γ', followed by layer normalization, giving r''_j.

Prediction heads. The numerical network applies a linear map $\mathbb{R}^{d'} \to \mathbb{R}$ to each numerical token r''_j, producing $\Delta_\chi(x, t, S)_j$. The categorical network applies a scalar head to every token and averages the outputs over columns, producing $\log h_\chi(x, t, S)$. For TABJUMP, every column is finite-state. Evaluating a scalar network separately at every candidate row $x^{j \leftarrow y}$ would require one forward pass per state of every column, for every row and sampling step. Instead, the network predicts the scores of all candidate rows in one pass. A linear map $\mathbb{R}^{d'} \to \mathbb{R}^{k_{\max}}$ is applied to each token r''_j, where $k_{\max}$ is the largest number of states of any column. Its output $\ell_\chi(x, t, S)_j[y] = \log h_\chi(x^{j \leftarrow y}, t, S)$ represents the guidance at the row $x^{j \leftarrow y}$, predicted from the current row x. Entries beyond the states of column j are ignored. All output heads are initialized to zero, so the untrained module leaves the base model and adds no corrections.

Column symmetry. All operations before the heads either act independently on each column or are the equivariant blocks of Lemma G.3. Permuting the columns together with their constraints therefore permutes r''. Hence Δ_χ and ℓ_χ are column equivariant, while $\log h_\chi$, obtained by averaging over columns, is column invariant. The conditioned model therefore preserves the symmetry of the base model in the sense of Definition G.1, with the permutation acting on S as well.

Hyperparameters. We use $d' = 48$, $L' = 2$, $H' = 4$, $\gamma' = 2$, $d_b = 8$, and $d_a = 8$, with no dropout. The tokenizer is frozen and shared with the base model. Both conditioning networks use the same architecture and hyperparameters. The TABJUMP network uses the same trunk with $k_{\max}$ outputs. Parameter counts are given in §H.5.

H.2 LOSSES

The module is trained using the conditional generator matching objective of Eq. 11. We draw $z \sim p^S_{\text{data}}$ while keeping the per-sample path and target of the unconditional model unchanged. Throughout expectations are taken with respect to,

$$z \sim p^S_{\text{data}}, \qquad t \sim \text{Unif}[0, 1], \qquad x_t \sim p_t(\cdot \mid z).$$

Diffusions. For a forward SDE $\mathrm{d}X_\tau = f(\tau, X_\tau)\,\mathrm{d}\tau + g(\tau)\,\mathrm{d}B_\tau$ (Song et al., 2020), a continuous-time evidence lower bound on the path KL divergence of the time reversal can be expressed as

$$\mathbb{E}\big[\|g(t)^2[\nabla \log p_t(x_t \mid z) - \nabla \log p_t(x_t)]\|^2\big]$$

where the second term is the score of the marginal, which adds to the drift of the reverse SDE (Song et al., 2021) and time is parametrized as $\tau = 1 - t$. Generally, $g(t)$ is chosen to be a diagonal

matrix in which case we will simply use the subscript j to talk about its j-th diagonal element. The numerical columns of TABDIFF-*sym* follow the variance-exploding case $f = 0$ where a close form solution is available to the SDE as

$$X_\tau = X_0 + \varsigma(\tau)\epsilon \quad \epsilon \sim \mathcal{N}(0, I)$$

Remark H.1. In this subsection, we will use the subscript j to denote only numerical columns, therefore all sums and subscripts involving j will pertain to numerical columns only.

Parameterizing this in reverse time $\tau = 1 - t$ yields that $x_{t,j} = z_j + \varsigma_j(t)\epsilon_j$ with $\epsilon_j \sim \mathcal{N}(0, 1)$ (§G.2) and $g_{t,j}^2 = -\mathrm{d}\varsigma_j(t)^2/\mathrm{d}t$ where $z_j = [X_0]_j$. In the noise parameterization (Ho et al., 2020) $\epsilon_\theta(x, t)_j = -\varsigma_j(t)\nabla \log p_t(x_t)_j$ of §G.1, the loss is rewritten as

$$\mathbb{E}\Big[\sum_{j \text{ numerical}} \frac{g_{t,j}^2}{\varsigma_j(t)^2}\big(\epsilon_j - \epsilon_\theta(x_t, t)_j\big)^2\Big]. \tag{128}$$

By Corollary 4.3, the j-th component of the drift of the conditioned reverse SDE can be expressed as $g_{t,j}^2\big(\partial_{x_j} \log p_t + \partial_{x_j} \log h\big)$, so the conditioned score of column j is now given as $s_\theta(x, t)_j + \partial_{x_j} \log h(t, x)$ and the conditioned noise prediction is $\epsilon_\theta(x, t)_j - \varsigma_j(t)\,\partial_{x_j} \log h(t, x)$. Adding the correction into the new drift, the conditional loss of Eq. 11 can be expressed as

$$\mathbb{E}\Big[\sum_j \frac{g_{t,j}^2}{\varsigma_j(t)^2}\Big(\epsilon_j - \epsilon_\theta(x_t, t)_j - \Delta_\chi(x_t, t, S)_j\Big)^2\Big], \tag{129}$$

where we write $\Delta_\chi(x_t, t, S)_j = -\varsigma_j(t)\,\partial_{x_j} \log h(t, x)$. Further, under other Bregman divergences D, Equation 129 generalizes the loss functions of Denker et al. (2024).

We make an important point here regarding the structure of $\Delta_\chi(x_t, t, S)$. Diffusions, in previous works (Ho et al., 2020), (Song et al., 2020), have seen many *equivalent* parametrizations of $\Delta_\chi(x_t, t, S)_j$, from scores, means, and noise predictors. We present the noise prediction loss function, but all of these are suitably generalized in Proposition 4.5.

Masked chains. Conditioning changes the rates of a jump process in general. For masked chains the total exit rate is unchanged, which allows a simpler parameterization. For the categorical columns of TABDIFF-*sym*, the target rates of column j are given as $Q_t(y, x)_j = \lambda_j(t)\, p_t(z_j = y \mid x)\, \mathbf{1}[x_j = \mathsf{M}]$ (§G.2). However, it turns out that we can simplify the rate of the conditioned chain $Q_t^S(y, x)_j = Q_t(y, x)_j\, h(t, x^{j \leftarrow y})/h(t, x)$. First, for a masked column j of a masked chain, following the time-reversal of Silverman & Kelly (1980) we get that

$$Q_t^f(x_j, y)_j p_t^S(y) = Q_t(y, x)_j \frac{h(t, x^{j \leftarrow y})}{h(t, x)} p_t^S(x_j)$$

where $Q_t^f(x_j, y)_j$ denotes the forward masking process. Since the per-sample paths remain the same, $Q_t^f(x_j, y)_j$ remains the same irrespective of conditioning. Crucially, the forward masking rate is independent of the starting category y. Therefore, the probability of observing the masked state $x_j = \mathsf{M}$ is independent of the starting distribution and $Q_t^f(x_j, y)_j = Q_t^f(x_j, y')_j = q_t^f(x_j) \forall y, y'$. This gives us that $p_t^S(x_j) = p_t(x_j)$. Hence,

$$Q_t(y, x)_j \frac{h(t, x^{j \leftarrow y})}{h(t, x)} = \frac{1}{p_t(x_j)} Q_t^f(x_j, y)_j p_t^S(y)$$

Summing, we get that

$$\sum_{y \neq x_j} Q_t(y, x)_j \frac{h(t, x^{j \leftarrow y})}{h(t, x)} = \frac{q_t^f(x_j)}{p_t(x_j)} \sum_{y \neq x_j} p_t^S(y)$$

Now since p_t^S is still a probability distribution, we get that $\sum_{y \neq x_j} p_t^S(y) = 1 - p_t(x_j)$. Hence,

$$\lambda_j^S(t) = \sum_{y \neq x_j} Q_t(y, x)_j \frac{h(t, x^{j \leftarrow y})}{h(t, x)} = \frac{q_t^f(x_j)}{p_t(x_j)}(1 - p_t(x_j)) = \sum_{y \neq x_j} Q_t(y, x)_j = \lambda_j(t)$$

where the second last equality uses the fact that the middle quantity is identical for the base process. This allows us to factor out the same rate factor out of both $Q_t^S(y,x)_j$ and $Q_t(y,x)_j$. Therefore, the new transition probabilities are given by $p_t^S(y,x)_j = p_t(y,x)_j \frac{h(t,x^{j\leftarrow y})}{h(t,x)}$. Summing up over all $y \neq x_j$ gives us that

$$h(t,x) = \sum_{y \neq x_j} p_t(y,x) h(t, x^{j\leftarrow y}) \tag{130}$$

Hence the conditioned process is the masked chain with the same rates and the reweighted endpoint posterior

$$p_{1|t}^S(z_j = y \mid x) = \frac{p_{1|t}(z_j = y \mid x)\, h(t, x^{j\leftarrow y})}{\sum_{y'} p_{1|t}(z_j = y' \mid x)\, h(t, x^{j\leftarrow y'})}. \tag{131}$$

Choosing the Bregman divergence D to be the generalized KL divergence yields an evidence lower bound on the path KL divergence (Holderrieth et al., 2025, App. D.2) and is given as

$$\mathbb{E}\Big[\sum_{j \text{ categorical}} \sum_{y \neq (x_t)_j} \Big(Q_t^\theta(y, x_t)_j - Q_t(y, x_t \mid z)_j \log Q_t^\theta(y, x_t)_j \Big) \Big]. \tag{132}$$

Since $Q_t(y, x \mid z)_j$ is nonzero only for $y = z_j$ and $\sum_{y \neq x_j} Q_t^\theta(y,x)_j = \lambda_j(t)\, \mathbf{1}[x_j = \mathsf{M}]$, the loss reduces to

$$\mathbb{E}\big[- \sum_{j \text{ categorical}} \lambda_j(t)\, \mathbf{1}[x_{t,j} = \mathsf{M}] \log \mu_\theta(x_t, t)_j[z_j] \big]$$

up to a constant in θ, which is the categorical loss used by TABDIFF. The conditioned loss on S is obtained from Proposition 4.5 by drawing z from p_{data}^S and replacing $\mu_\theta[z_j]$ by the conditioned posterior of Eq. 131,

$$\mathbb{E}\Big[- \sum_{j \text{ categorical}} \lambda_j(t)\, \mathbf{1}[x_{t,j} = \mathsf{M}] \log \Big(\mu_\theta(x_t, t)_j[z_j] \, \frac{h(t, x_t^{j\leftarrow z_j})}{h(t, x_t)} \Big) \Big] \tag{133}$$

up to a constant. TABDIFF parameterizes the logits of μ_θ directly, and we leave them frozen. The identity of Eq. 130 need not be enforced when h is parameterized at the rows $x_t^{j\leftarrow y}$ and so we choose to instead substitute $h(t, x_t)$ with Eq. 130 giving the loss as

$$\begin{aligned} \mathbb{E}\Big[\sum_{j \text{ categorical}} \lambda_j(t)\, \mathbf{1}[x_{t,j} = \mathsf{M}] \Big(& - \log \mu_\theta(x_t, t)_j[z_j] - \log h_\chi(x_t^{j\leftarrow z_j}, t, S) \\ & + \log \sum_y \mu_\theta(x_t, t)_j[y]\, h_\chi(x_t^{j\leftarrow y}, t, S) \Big) \Big], \end{aligned} \tag{134}$$

where h has been replaced by its neural approximation h_χ. In terms of the base logits ℓ_j of column j, the bracket equals $-\ell_j[z_j] - \log h_\chi(x_t^{j\leftarrow z_j}, t, S) + \log \sum_y e^{\ell_j[y] + \log h_\chi(x_t^{j\leftarrow y}, t, S)}$ up to the normalizer of ℓ_j. We therefore minimize the final loss given as

$$\begin{aligned} \mathbb{E}\Big[\sum_{j \text{ categorical}} \lambda_j(t)\, \mathbf{1}[x_{t,j} = \mathsf{M}] \Big(& - \log h_\chi(x_t^{j\leftarrow z_j}, t, S) \\ & + \log \sum_y e^{\ell_j[y] + \log h_\chi(x_t^{j\leftarrow y}, t, S)} \Big) \Big], \end{aligned}$$

Per-column structure. From (Holderrieth et al., 2025, Prop. 5), both objectives are sums over columns, where each term may depend on the full row. Together with the symmetry of §H.1, this makes the loss invariant to permuting the columns together with their constraints. For TABDIFF-*sym* the numerical and categorical losses are both added up equally. For TABJUMP there is a single loss.

H.3 TRAINING

Training constraint sets. The training constraint sets are those of §I.3. For each constraint set, the training rows that satisfy it define the empirical distribution p_{data}^S used in Eq. 11.

Algorithm 3 Training the conditioning module

Require: frozen base model θ; training constraints and their supports; steps T; batch size b; constraints per step q

initialize the conditioning networks χ with zero output heads; $\bar{\chi} \leftarrow \chi$
construct the fixed selection set
for step $= 1, \dots, T$ **do**
    sample q constraints $S^{(1)}, \dots, S^{(q)}$ uniformly
    for each $S^{(i)}$ **do**
        sample b/q satisfying rows z with replacement
        sample $t \sim \mathrm{Unif}[0, 1]$ for each row and generate x_t using the base process
        evaluate the frozen base model $\mu_\theta(x_t, t)$
        compute Eq. 129 using $\Delta_\chi(x_t, t, S^{(i)})$
        evaluate $\log h_\chi$ at the rows $x_t^{j \leftarrow y}$, for every masked categorical column j and every $y \in E_j$, and compute Eq. 134
    end for
    update χ with AdamW and update the EMA parameters $\bar{\chi}$
    if step $> 3{,}000$ and step is a multiple of 250 **then**
        evaluate $\bar{\chi}$ on the selection set and save it if the loss is the lowest so far
    end if
end for
return the saved $\bar{\chi}$

Procedure. Algorithm 3 summarizes training. At each step, we sample q constraints and b/q satisfying rows for each constraint, with replacement. For every row, we sample t, generate x_t using the base process, and evaluate the frozen base model once. We then optimize Eqs. 129 and 134. The same recipe is used for TABJUMP, with $t \sim \mathrm{Unif}[0, 1 - \varepsilon_{\mathrm{cut}}]$ and x_t drawn from its masked path (§G.4). We use AdamW with learning rate 10^{-3}, weight decay 10^{-4}, batch size $b = 1024$, $q = 8$ constraints per step, EMA decay 0.997, and $10{,}000$ training steps. Before training, we construct a fixed selection set with 32 noised rows per training constraint and cache their frozen base-model outputs. Every 250 steps after step $3{,}000$, we evaluate the EMA parameters on this set and retain the checkpoint with the lowest loss.

H.4 SAMPLING

Algorithm 4 gives the conditional sampler. At each of the $N = 50$ steps, the frozen base model is evaluated once. For numerical columns, Δ_χ is added to the base denoiser and the corrected score is used in the probability-flow ODE. For categorical columns, $\log h_\chi$ is evaluated at every candidate row $x^{j \leftarrow y}$ of every masked column j, and the original unmasking schedule is retained with the probabilities replaced by Eq. 131. Both corrections are applied with unit weight. Before being added to the denoiser, the numerical correction is clipped to $[-5, 5]$ on the transformed scale. This is a numerical safeguard used by the implementation. For TABJUMP, the guided posterior enters the Euler step of Algorithm 2 in place of $\mu_{\bar{\theta}}$, with the same jump probabilities, since the exit rates are unchanged. When a sampled numerical bin lies only partly inside the constraint interval, the decoded value is drawn uniformly from the part of the bin inside the interval, so that every generated row satisfies the constraint. For TABBYFLOW, no learned conditioning module is required. The training-free conditioner of §F.1 replaces the endpoint means and leaves the ODE solver unchanged.

H.5 COST

The numerical correction requires one module evaluation per row and sampling step, as does the TABJUMP network. For TABDIFF-*sym*, the categorical module must be evaluated once for every candidate value of every masked column. A step therefore requires up to $\sum_{j\,\mathrm{categorical}} |E_j|$ categorical-module evaluations per row. This is 67 on Shoppers, 104 on Adult, and $2{,}387$ on Diabetes, where three diagnosis columns contain 717, 749, and 790 categories.

The conditioning module is small, with width 48 and two Transformer blocks; Table 6 gives its parameter counts next to those of the base models. Its categorical cost nevertheless grows linearly

Algorithm 4 Sampling with the conditioning module

Require: frozen base model θ; trained module $\bar{\chi}$; constraint S; steps N
1: draw x from the base source distribution
2: **for** $i = 0, \dots, N-1$ with $t = t_i$ **do**
3: evaluate the frozen base model $\mu_\theta(x, t)$
4: numerical columns: $\mu_j^S \leftarrow \mu_\theta(x,t)_j + \mathrm{clip}\big(\Delta_{\bar{\chi}}(x, t, S)_j, -5, 5\big)$
5: advance x^{num} using the probability-flow ODE with μ^S
6: categorical columns: for every masked j and $y \in E_j$, set $\log h_j[y] \leftarrow \log h_{\bar{\chi}}(x^{j \leftarrow y}, t, S)$
7: $\mu_j^S \leftarrow \mathrm{softmax}_y\big(\log \mu_\theta(x,t)_j[y] + \log h_j[y]\big)$
8: advance x^{cat} using the base unmasking step with μ_j^S
9: **end for**
10: **return** x after the base model's decoding

Table 6: Parameter counts **in millions** of the frozen base model (from Table 3) and of the conditioning module trained on top of it. For TABDIFF-*sym* the module count is the total over its numerical and categorical networks; TABJUMP has a single network, since binning the numerical columns removes the numerical branch. The frozen base tokenizer is shared and not counted.

	TABDIFF-*sym*		TABJUMP	
Dataset	base	conditioning module	base	conditioning module
SHOPPERS	2.94	0.14	3.54	0.08
MAGIC	2.90	0.14	3.97	0.08
BEIJING	2.86	0.14	3.05	0.08
DEFAULT	3.05	0.15	4.35	0.08
ADULT	2.86	0.14	3.04	0.08
NEWS	3.82	0.14	7.02	0.09
DIABETES	4.12	0.15	4.06	0.11
FB-COMMENTS	3.65	0.15	6.08	0.09

with total cardinality and dominates for high-cardinality tables. Table 9 reports sampling times per $1{,}000$ rows. Training takes about 40 minutes per dataset on one GPU and several hours on Diabetes. This linear dependence on categorical cardinality is the main computational limitation of the Doob correction for masked chains. The direct head of TABJUMP avoids it at the price of predicting $\log h$ at every row $x^{j \leftarrow y}$ from the current row alone.

I DATASETS, SPLITS, AND CONSTRAINT SETS

This section describes the datasets used in §6 (§I.1), the row splits used to train the unconditional models (§I.2), and the suite of constraint sets on which conditioning is trained and evaluated (§I.3).

I.1 DATASETS

We use eight tables: the six benchmark tables of Zhang et al. (2024), namely SHOPPERS, MAGIC, DEFAULT, NEWS, BEIJING, and ADULT; DIABETES from Shi et al. (2025); and the larger FB-COMMENTS table of Gorishniy et al. (2021). Each table comes with a prediction task and a designated target column, which we treat as an ordinary column of its type: binary classification of `Revenue` (SHOPPERS), `class` (MAGIC), `default payment next month` (DEFAULT), `income` (ADULT), and `readmitted` (DIABETES), and regression of `shares` (NEWS), `pm2.5` (BEIJING), and the comment count (FB-COMMENTS). Table 7 lists the sizes, sorted by number of rows. DIABETES contains three diagnosis-code columns with 717, 749, and 790 distinct values, making it the high-cardinality case for the categorical conditioning of Appendix H.

Dataset	Rows	Num.	Cat.	Task	Train	Val.	Test
SHOPPERS	12,330	10	8	classification	9,987	1,110	1,233
MAGIC	19,019	10	1	classification	15,405	1,712	1,902
DEFAULT	30,000	14	10	classification	24,300	2,700	3,000
NEWS	39,644	46	2	regression	32,111	3,568	3,965
BEIJING	41,757	7	5	regression	33,822	3,759	4,176
ADULT	48,842	6	9	classification	29,304	3,257	16,281
DIABETES	101,766	9	28	classification	61,059	20,353	20,354
FB-COMMENTS	197,080	37	15	regression	157,638	19,722	19,720

Table 7: Datasets, sorted by number of rows. Num. and Cat. count numerical and categorical columns, including the target column. Train, Val., and Test are the row splits of §I.2.

I.2 DATASET SPLITS FOR UNCONDITIONAL TRAINING

We use the published train/test splits of Zhang et al. (2024): a $90/10$ split for SHOPPERS, MAGIC, DEFAULT, NEWS, and BEIJING, and the official test file for ADULT. For DIABETES we use the train/validation/test split of Shi et al. (2025), and for FB-COMMENTS that of Gorishniy et al. (2021). Some training procedures need a separate validation split: TABPC uses one for early stopping and learning-rate decay (§G.2), and TABSYN and TABSYNFLOW use one to select their autoencoder (§G.2). Where the published split has no validation part, we therefore hold out 10% of the published training rows as a validation split, which gives the $81\%/9\%/10\%$ layout of Table 7. The held-out rows are chosen by a seeded permutation, following the rule of TABPC as follows. The seed starts at 1234 and is incremented until the remaining training rows contain every level of every categorical column. The test split is never touched. We refer to the three parts as the training, validation, and test splits, and to training and validation together as the published training rows. Methods whose training procedure does not use a validation split, that is TABDIFF, TABBYFLOW, and TABJUMP, are trained on the published training rows, exactly as in their reference implementations. TABPC is trained on the training split with the validation split held out. TABSYN and TABSYNFLOW train their autoencoder on the training split and select it on the validation split, and train their latent model on the published training rows. The column statistics used to build constraint sets in §I.3 are computed on the training split only.

I.3 CONSTRAINT SETS FOR TRAINING AND EVALUATING CONDITIONING

The conditioning module of Appendix H is trained on many constraint sets and evaluated on constraint sets that it has not seen. We describe how these sets are generated.

A constraint set is a product $S = \prod_j S_j$. For a constrained numerical column, $S_j = [a_j, b_j]$. For a constrained categorical column, $S_j \subseteq E_j$. For an unconstrained column, $S_j = E_j$. We write $J_S = \{j : S_j \neq E_j\}$ for the constrained columns and $k = |J_S|$ for the *arity*. The *selectivity* of S is the fraction of rows of a given split that satisfy it. Its *predicate type* is numerical if all constrained columns are numerical, categorical if all are categorical, and mixed otherwise.

Candidate grid. We generate constraint sets at seven target selectivities, $\varpi \in \{40, 25, 10, 5, 2, 1, 0.5\}\%$, ten arities $k \in \{1, \ldots, 10\}$, and three predicate types. We generate eight candidates for each combination. This gives at most $7 \times 10 \times 3 \times 8 = 1{,}680$ candidates per dataset. Combinations that a dataset cannot support are skipped. For example, MAGIC cannot form a categorical constraint of arity 3.

Each candidate is built in three steps. We first choose the constrained columns. We then assign a target fraction of rows to each predicate. Finally, we jointly tighten or loosen the predicates until the selectivity of the full constraint set is close to ϖ. Algorithm 5 summarizes the procedure.

Column choice. For a numerical or categorical candidate, all k columns are chosen from that type. For a mixed candidate, we choose $\max(1, \lfloor k/2 \rfloor)$ numerical columns and use categorical columns for the rest. A mixed candidate with $k = 1$ contains one numerical predicate.

To spread the constraints across columns, we keep track of how often each column has been used. The k columns are sampled without replacement from the half of the eligible columns used least often so far.

Target fractions. Each predicate j is assigned a target fraction ϖ_j such that $\prod_{j \in J_S} \varpi_j = \varpi$. Under the *equal* scheme, all predicates receive $\varpi_j = \varpi^{1/k}$. Under the *skewed* scheme, one randomly chosen predicate receives $\varpi^{1/2}$ and each of the remaining $k-1$ predicates receives $\varpi^{1/(2(k-1))}$. For $k = 1$, both schemes give $\varpi_1 = \varpi$. The equal and skewed schemes alternate across the eight candidates of each combination.

Predicates. For a numerical column, a predicate with target fraction ϖ_j is an interval containing approximately that fraction of the training rows. We draw $u_j \sim \mathrm{Unif}[0, 1)$ and use the interval between the empirical quantiles at $u_j(1 - \varpi_j)$ and $u_j(1 - \varpi_j) + \varpi_j$. The quantiles are computed on the training split.

For a categorical column, with probability 0.7 we use a single value whose training frequency is closest to ϖ_j. Otherwise, we use the smallest set of most frequent values whose cumulative training frequency reaches ϖ_j.

Adjustment to the target selectivity. The predicates are constructed one column at a time, but columns can be dependent. Their joint selectivity can therefore differ from the target ϖ. We adjust all predicates together using a common exponent $\eta > 0$. For a given η, predicate j is rebuilt using the fraction ϖ_j^{η}, while keeping its random choices fixed. Increasing η makes the predicates tighter, while decreasing it makes them wider.

We search for η by 30 steps of bisection on $[0.05, 12]$. We keep the value whose selectivity on the training split is closest to ϖ.

Acceptance and selectivity bands. An adjusted candidate is accepted if at least 50 training rows satisfy it and if its selectivity is at most 50% on both the training and test splits. Accepted constraint sets are grouped by their realized training selectivity into the bands 0.5–1%, 1–2%, 2–5%, 5–10%, 10–25%, and 25–50%. The original target ϖ is retained as a generation label. For every accepted constraint set, we store the satisfying rows from each split.

Training and test constraint sets. Candidates with identical predicates are merged. The remaining constraint sets are split $80/20$ into training and test constraint sets with a fixed seed. The split is stratified by target selectivity, arity, and predicate type. A stratum containing only one constraint set is assigned to training.

The conditioning module is trained only on the training constraint sets. All conditional results in §6 use the test constraint sets, which are never used to train the module. After acceptance and duplicate removal, each dataset contains between 673 and $1{,}521$ distinct constraint sets, as reported in Table 8.

The implementation also generates a separate set of full-arity constraint sets with one predicate on every column. These are not used in this paper.

Dataset	Accepted	Distinct	Training	Test
SHOPPERS	1,291	1,243	969	274
MAGIC	712	673	505	168
DEFAULT	1,554	1,521	1,172	349
NEWS	838	742	563	179
BEIJING	877	791	602	189
ADULT	1,082	1,054	825	229
DIABETES	1,070	1,062	826	236
FB-COMMENTS	1,231	1,210	934	276

Table 8: Constraint suites, in the order of Table 7. Accepted is the number of candidates, out of at most $1{,}680$, that pass the acceptance test of §I.3; Distinct is the number remaining after merging identical predicate sets; Training and Test give their split into training and test constraint sets.

Algorithm 5 Generation of the constraint suite of a dataset

Require: training-split rows and their column statistics; target selectivities $\mathcal{B}$; arities $1, \ldots, 10$; predicate types; candidates per combination c

for each target selectivity $\varpi \in \mathcal{B}$, arity k, and predicate type **do**
  if the dataset has enough columns of the required types **then**
    for $i = 1, \ldots, c$ **do**
      choose k columns of the required types from the least-used half of the eligible columns
      assign shares $(\varpi_j)_{j \in J_S}$ with $\prod_j \varpi_j = \varpi$, equal for even i and skewed for odd i
      draw and fix the interval positions u_j and the single-value or set choices
      find $\eta \in [0.05, 12]$ by bisection so that the predicates built from the shares ϖ_j^η give a training-split selectivity closest to ϖ
      if at least 50 training-split rows satisfy the constraint set and its selectivity is at most 50% on the training and on the test split **then**
        accept it; record its predicates, selectivity band, and satisfying rows of each split; update the column-usage counts
      end if
    end for
  end if
end for
merge identical constraint sets
split the distinct constraint sets $80/20$ into training and test constraint sets, stratified by target selectivity, arity, and predicate type
return the training and test constraint sets

I.4 Constrained generation baselines

We compare against imputation, penalty guidance, and LLM-based conditioning. All baselines were run on one V100 per job, with the adaptations below.

DiffPuter. DiffPuter (Zhang et al., 2025) performs diffusion-based imputation with a 200-step DDPM and an MLP denoiser. Observed cells are reintroduced at every reverse step. Its checkpoint is shared with Harpoon. We use the implementation provided in the Harpoon codebase. Since its interface accepts values rather than predicates, we sample allowed categorical values from their restricted training marginals. Numerical intervals are left unconditioned. We use the same predicate mapping for GReaT.

Harpoon. Harpoon (Shankar et al., 2026) applies training-free penalty guidance to the DDPM. Each reverse step is followed by $x \leftarrow x - \eta \nabla_x L(\hat{x}_0(x))$, using the reference strength $\eta = 0.2$.

We also ported this rule to the frozen TabDiff model to compare penalty and Doob guidance on the same generator. However, its fixed penalty step is incompatible with the shrinking diffusion increments of TabDiff's Heun sampler. The port concentrates numerical samples near interval boundaries and provides insufficient logit shifts for rare categorical constraints. Its joint hit rates fall to 0.12–0.28, compared with 0.59 for the original DDPM implementation, which we therefore report.

GReaT. GReaT (Borisov et al., 2022) fine-tunes GPT-2 on serialized rows and conditions through prompted imputation. We train for 100 epochs per dataset, use schema-defined column types to avoid invalid numerical outputs for integer-coded categories, and implement batched prompting with retries. Batching is approximately $60\times$ faster without a measurable change in outputs. With our predicate mapping, GReaT satisfies categorical constraints almost always, but numerical intervals only at their base rate, since its imputation interface cannot represent them.

Our conditioners. The learned guides use guidance strength 1, without parameter sweeps, and checkpoints selected by loss evaluated with EMA weights (§H.3). The TABBYFLOW-*sym* conditioner requires no additional training. We also report unconditional samplers, whose expected hit rates equal constraint selectivity.

Runtime. Table 9 reports generation time per 1,000 rows, excluding training. DIFFPUTER requires a single ancestral pass, whereas HARPOON adds a backward pass per step. The TABDIFF-*sym* guide is expensive on DIABETES because it evaluates candidate states separately; TABJUMP predicts them in one pass and is approximately $80\times$ faster there (§H.5). GREAT is slowest on NEWS, which has the longest serialized rows.

Table 9: Mean wall-clock seconds per 1,000 generated rows for one constraint set, on one V100. Training time excluded.

	SHOPPERS	MAGIC	DEFAULT	ADULT	NEWS	BEIJING	DIABETES	FB-COMMENTS
DIFFPUTER	0.4	0.4	0.4	0.4	0.4	0.4	0.6	0.4
HARPOON	1.4	1.2	1.4	1.5	1.2	1.4	3.3	1.4
GREAT	7.6	5.0	15.2	8.1	76.0	4.2	21.0	44.2
TABBYFLOW-*sym* (closed form)	1.5	0.9	1.8	1.4	3.0	1.1	3.2	6.8
TABDIFF-*sym* + guide	7.1	1.0	9.7	9.0	4.1	5.1	216.0	53.8
TABJUMP + guide	1.4	0.9	1.7	1.2	3.2	1.0	2.7	3.4

J EXPERIMENTAL METHODOLOGY

There has been extensive work (Scassola et al., 2026), Jiang et al. (2026) on trying to characterize the properties one desires from tabular generation. In line with this, we briefly describe other metrics we use for a holistic evaluation of our methodology.

J.1 A DISCUSSION ON METRICS EVALUATED

J.1.1 FIDELITY METRICS

α-PRECISION and β-RECALL (Alaa et al., 2022). α-PRECISION measures how close the synthetic data is to the support of the real data while β-RECALL measures how close is the real data to the support of the synthetic data. Following the discussion from Scassola et al. (2026), we use the *One Class* embedding following Alaa et al. (2022).

C2ST (Lopez-Paz & Oquab, 2016). The classifier two-sample test (C2ST) trains a logistic regression to distinguish real from synthetic data. As reported in Scassola et al. (2026), the regular LR classifier is too weak to report any meaningful stratification of fidelity, and its performance therefore remains saturated. We report it in the appendix for completeness.

C2ST-XGB Scassola et al. (2026). The classifier test now uses a much stronger XGBoost classifier following Scassola et al. (2026), who show that XGBoost offers a much clearer stratification of model performance.

SHAPE & TREND (Dat, 2025) SHAPE, implemented in SDMetrics (Dat, 2025), measures how well synthetic data captures the univariate marginal distributions of the real data, while TREND measures how well synthetic data captures pairwise column dependencies. We use the implementation as presented in Shi et al. (2025).

J.1.2 PRIVACY METRICS

DCR. We report the Distance to Closest Record (DCR) score following the protocol of Shi et al. (2025), which measures the probability that a synthetic sample's nearest neighbor is drawn from the training set rather than a held-out test set. This is converted into DCR Balance by taking its distance from 0.5 and expressing it as a percentage.

EMR We also report the Exact Match Rate of rows, as a sanity check against memorization of the training data.

J.1.3 UTILITY METRICS

MLE. We evaluate Machine Learning Efficiency (MLE) following the protocol and implementation of Shi et al. (2025): a designated column is masked out and a downstream classifier or regressor is trained on the synthetic rows, and then evaluated on the held-out test rows.

J.2 PROTOCOL FOR UNCONDITIONAL EVALUATION

Each model is trained once and sampled with multiple seeds. We report the mean across seeds. Following Zhang et al. (2024); Shi et al. (2025), SHAPE, TREND, and C2ST use training-sized synthetic samples and the published training split as reference. MLE trains predictors on training-sized synthetic samples and evaluates them on the test split. α-precision, β-recall, and DCR balance use test-sized synthetic samples and the test split, with an equally sized training reference for DCR balance. EMR compares test-sized synthetic samples against the full training split.

J.3 PROTOCOL FOR CONDITIONAL EVALUATION

For each dataset, we split the constraint suite of §I.3 $80/20$ into training and test constraints, stratified by target selectivity, arity, and modality mix. We evaluate only on test constraints. For each constraint S, every method generates $1{,}000$ rows per seed across five sampling seeds, using 50 reverse steps. hit rate is the fraction of generated rows satisfying S. Following Shankar et al. (2026), we compute fidelity metrics using all generated rows and compare them against published training rows satisfying S, subsampling the larger set to match the with a minimum size of 50 rows to maintain statistical significance. This controls the dependence of C2ST on sample size. We report means across seeds and constraints, grouped by realized training selectivity and arity from Figures 3 to 8.

K DETAILED EXPERIMENTAL RESULTS

K.1 UNCONDITIONAL EVALUATION

We report our observations here. In consistency with previous works, Tables 10-18 present below record the evaluation of SHAPE, TREND, α-PRECISION, β-RECALL, C2ST, C2ST-XGB, DCR and EMR on all 8 datasets on all unconditional baselines:

1. **Symmetric v/s Non-symmetric counterparts:** On SHAPE, TABDIFF-*sym* and TABBYFLOW-*sym* outperform their counterparts TABDIFF and TABSYN on 7 out of 8 datasets on SHAPE. and 6 out of 8 datasets on TREND. Performance on α-PRECISION and β-RECALL remains virtually the same. The individual improvements are modest since the metrics are saturated.
2. **Performance on C2ST-XGB.** Across all but one datasets, TABDIFF-*sym* and TABBYFLOW-*sym* outperform TABDIFF and TABBYFLOW, and further TABPC which beat them before, is now beaten by the symmetric variants. Notably, TABJUMP produces the highest scores on *all* datasets, often far exceeding its nearest competitor.
3. **Data Space v/s Latent Space Methods:** Under most fidelity metrics, TABDIFF and TABBYFLOW almost always outperform their latent space counterparts TABSYN and TABSYNFLOW.
4. **DCR Balance.** This highlights that the methods are not simply memorizing the training data. Almost all methods show similar numbers, affirming that there is little memorization of training data.
5. **Saturation of Existing Benchmarks:** As reported by Scassola et al. (2026), SHAPE, TREND, C2ST, α-PRECISION, and β-RECALL, MLE are all highly saturated with competitive performance by most methods across all datasets. C2ST can further be gamed by even fully factorized models as seen in Scassola et al. (2026), and so we do not comment on it.
6. **EMR:** As a final sanity check, we try to evaluate direct matches with the data. All methods show negligible memorization, except on SHOPPERS, but TABJUMP in general has higher match rate possibly due to its exactness of decoding.

K.2 Conditional Evaluation

Figures 3 to 8 present below record the hit rate, SHAPE and C2ST-XGB of all conditional baselines across all 8 datasets. Our motivation for studying conditional generation was to draw provably from the true conditional distribution. In light of this, we opt to study the hit rate which measures whether we satisfy the constraints, and SHAPE and C2ST-XGB, which describe the fidelity of the data drawn from the conditional distribution. We report our observations here.

1. **Variations across Query arity.** Metrics show a less pronounced variation as the query arity increases. This in contrast to selectivity which shows a marked increase in numbers as the selectivity is relaxed.
2. **Performance across arity:** TABJUMP still continues to be the best performer on SHAPE and C2ST-XGB, as arity is changed across *all* datasets. TABDIFF here comes a close second in SHAPE on 7 and on C2ST-XGB on 6 out of the 8 datasets. HARPOON and TABBYFLOW while achieving high hit rates, fail to maintain fidelity. The performance of HARPOON is especially bad on DIABETES where although it achieves high constraint satisfaction, its SHAPE is 50% that of the top performers.
3. **Performance across query selectivity.** This largely carries over from § 6.2 with most of the observations remaining identical. There is a marked increase in hit rate and SHAPE as the selectivity increases, which is not observed across arity. This suggests that guidance is good when trying to reach possibly low-dimensional unlocalized spaces, but is significantly harder to reach high-dimensional but extremely localized spaces.

Table 10: Shape Error in (%); lower is better).

	TABDIFF		TABBYFLOW		TABSYN		TABSYNFLOW		TABPC	TABJUMP
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	Shape error (%) ↓									
SHOPPERS	1.36	1.19	1.45	0.78	1.66	1.25	1.85	1.11	**0.75**	2.75
MAGIC	0.84	0.86	0.84	**0.75**	0.82	0.86	0.98	1.00	0.88	0.94
DEFAULT	1.32	0.75	1.12	0.65	1.22	0.77	1.40	0.99	**0.54**	1.08
NEWS	2.56	1.08	2.56	0.92	3.86	1.27	2.08	1.11	**0.57**	0.99
BEIJING	1.06	0.82	0.91	0.62	5.72	0.76	1.25	0.89	**0.56**	0.94
ADULT	0.72	0.56	0.79	**0.54**	2.83	0.59	0.95	0.73	0.55	1.07
DIABETES	1.00	0.61	2.07	1.16	5.87	0.78	2.41	0.87	**0.56**	0.57
FB-COMMENTS	0.73	0.73	0.46	0.34	0.74	0.33	0.98	0.61	**0.32**	0.62

Table 11: Trend error (%; lower is better).

	TABDIFF		TABBYFLOW		TABSYN		TABSYNFLOW		TABPC	TABJUMP
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	Trend error (%) ↓									
SHOPPERS	1.90	1.67	2.23	**1.42**	2.39	2.25	2.78	2.33	1.66	3.18
MAGIC	0.89	1.19	0.83	1.12	0.85	1.03	**0.82**	1.26	1.25	1.66
DEFAULT	4.61	3.61	5.08	**2.13**	3.01	3.81	2.70	5.08	5.56	2.72
NEWS	1.85	**1.06**	1.60	1.18	2.35	1.30	1.61	1.27	1.87	1.29
BEIJING	2.56	2.41	2.23	**1.79**	7.33	2.58	2.87	2.61	2.13	1.95
ADULT	1.60	**1.22**	1.76	1.23	4.88	1.58	2.29	1.93	1.41	1.80
DIABETES	2.53	**1.63**	4.20	2.59	13.72	2.52	5.27	2.43	1.85	1.90
FB-COMMENTS	2.07	2.25	**1.27**	1.44	2.34	2.29	1.98	2.67	1.84	1.34

Table 12: C2ST-XGBoost similarity (higher is better).

	TabDiff		TabbyFlow		TabSyn		TabSynFlow		TabPC	TabJump
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	C2ST-XGBoost ↑									
Shoppers	0.73	0.82	0.75	0.80	0.65	0.69	0.61	0.65	**0.91**	0.85
Magic	0.76	0.73	0.79	0.80	0.75	0.75	0.69	0.61	**0.88**	0.75
Default	0.60	0.72	0.51	0.46	0.43	0.51	0.32	0.42	0.27	**0.75**
News	0.03	0.32	0.13	0.24	0.07	0.37	0.10	0.35	0.12	**0.57**
Beijing	0.78	0.83	0.83	0.74	0.38	0.64	0.63	0.57	0.57	**0.95**
Adult	0.84	0.88	0.85	0.87	0.59	0.79	0.74	0.76	0.87	**1.00**
Diabetes	0.73	0.99	0.49	0.70	0.15	0.78	0.37	0.76	0.79	**1.00**
FB-comments	0.61	0.65	0.10	0.17	0.04	0.27	0.03	0.30	0.00	**0.84**

Table 13: C2ST-Logistic similarity (higher is better).

	TabDiff		TabbyFlow		TabSyn		TabSynFlow		TabPC	TabJump
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	C2ST-Logistic ↑									
Shoppers	0.987	0.988	0.977	0.996	0.960	0.971	0.930	0.972	**0.999**	0.888
Magic	0.997	0.997	0.997	0.997	0.998	0.997	0.994	0.991	**0.999**	0.988
Default	0.970	0.991	0.981	0.997	0.971	0.996	0.978	0.986	**0.999**	0.957
News	0.899	0.982	0.915	0.987	0.837	0.955	0.950	0.967	**0.996**	0.957
Beijing	0.979	0.989	0.995	0.999	0.871	0.991	0.971	0.985	**0.999**	0.961
Adult	0.982	0.994	0.979	0.997	0.889	0.998	0.983	0.988	**0.998**	0.950
Diabetes	0.957	**1.000**	0.623	0.811	0.259	0.874	0.552	0.860	**1.000**	**1.000**
FB-comments	0.988	0.984	0.987	0.997	0.964	0.987	0.953	0.977	**0.998**	0.974

Table 14: DCR balance $100|r - 0.5|$ (lower is better).

	TabDiff		TabbyFlow		TabSyn		TabSynFlow		TabPC	TabJump
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	DCR balance ↓									
Shoppers	5.86	5.73	7.62	4.47	4.85	**4.15**	5.32	4.53	5.93	6.23
Magic	2.37	**1.85**	3.84	3.22	2.52	2.47	2.25	2.50	3.86	2.74
Default	**0.81**	1.00	0.81	1.34	1.36	1.31	1.04	1.13	1.10	1.20
News	13.12	10.89	12.34	10.64	11.18	**10.22**	10.72	10.28	12.09	11.11
Beijing	1.78	1.90	3.92	1.62	**0.72**	0.84	1.42	0.90	1.94	2.43
Adult	4.00	3.41	6.21	**0.41**	0.61	0.83	0.45	0.77	2.61	5.82
Diabetes	3.97	3.04	5.82	3.79	**1.26**	1.48	1.58	1.32	2.99	3.12
FB-comments	1.04	1.39	**0.33**	0.93	1.25	1.24	0.95	1.36	0.94	0.48

Table 15: Exact-match rate in basis points (lower is better).

	TabDiff		TabbyFlow		TabSyn		TabSynFlow		TabPC	TabJump
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	Exact match (bp) ↓									
Shoppers	136.25	151.26	259.12	153.28	105.43	98.54	119.22	**92.46**	156.93	304.54
Magic	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Default	11.50	10.83	11.17	11.00	11.67	10.17	6.33	6.17	**4.00**	27.50
News	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Beijing	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.75
Adult	0.03	0.00	0.00	0.00	0.03	0.00	0.00	0.03	0.03	1.07
Diabetes	0.00	0.02	0.56	0.02	0.00	0.00	0.00	0.00	0.00	9.16
FB-comments	2.97	3.17	0.05	0.03	0.00	0.28	0.00	0.74	0.00	73.66

Table 16: One-class α-Precision (higher is better).

	TabDiff		TabbyFlow		TabSyn		TabSynFlow		TabPC	TabJump
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	α-Precision ↑									
Shoppers	0.91	0.92	**0.94**	0.91	0.90	0.91	0.92	0.89	0.89	0.92
Magic	0.85	0.84	0.84	0.84	0.86	0.86	**0.87**	0.85	0.87	0.79
Default	0.95	0.95	**0.97**	0.96	0.96	0.96	0.96	0.96	0.95	0.94
News	0.12	0.08	0.08	0.09	**0.13**	0.08	0.11	0.08	0.10	0.10
Beijing	0.94	0.97	0.96	**0.99**	0.99	0.94	0.90	0.95	0.97	0.98
Adult	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99
Diabetes	0.99	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99
FB-comments	0.67	0.63	0.45	**0.73**	0.47	0.68	0.47	0.66	0.59	0.67

Table 17: One-class β-Recall (higher is better).

	TabDiff		TabbyFlow		TabSyn		TabSynFlow		TabPC	TabJump
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	β-Recall ↑									
Shoppers	0.48	0.48	**0.50**	0.48	0.48	0.48	0.49	0.47	0.49	0.48
Magic	0.48	0.47	0.47	0.47	0.48	0.47	**0.49**	0.47	0.47	0.45
Default	0.49	0.49	**0.50**	0.49	0.49	0.49	0.49	**0.50**	0.48	0.48
News	0.07	0.03	0.01	0.02	0.07	0.03	0.07	0.01	0.07	0.07
Beijing	**0.47**	0.45	0.46	0.44	0.42	0.45	**0.47**	0.45	0.43	0.45
Adult	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49
Diabetes	0.49	**0.50**	0.48	0.49	0.42	0.48	0.46	0.48	0.49	**0.50**
FB-comments	0.65	0.62	0.45	**0.69**	0.42	0.65	0.42	0.64	0.56	0.65

Table 18: Machine-learning efficiency measured by AUC for classification and RMSE for regression datasets marked †.

	TabDiff		TabbyFlow		TabSyn		TabSynFlow		TabPC	TabJump
	orig.	*sym*	orig.	*sym*	orig.	*sym*	orig.	*sym*		
	MLE (AUC / RMSE†)									
Shoppers	0.921	**0.924**	0.914	0.923	0.922	0.914	0.921	0.907	0.898	0.905
Magic	0.937	0.933	0.937	0.938	0.937	**0.939**	0.934	0.932	0.935	0.924
Default	0.759	0.762	0.752	**0.763**	0.759	0.757	0.759	0.756	0.748	0.762
News †	0.862	0.855	0.852	**0.846**	0.862	0.856	0.855	0.854	0.877	0.853
Beijing †	0.562	0.575	**0.535**	0.605	0.689	0.642	0.601	0.663	0.655	0.583
Adult	0.913	0.913	0.911	0.912	0.905	0.909	0.908	0.908	0.915	**0.923**
Diabetes	0.693	**0.695**	0.693	0.694	0.661	0.686	0.676	0.687	0.691	0.695
FB-comments †	0.471	0.472	**0.466**	0.473	0.476	0.473	0.479	0.470	0.545	0.482

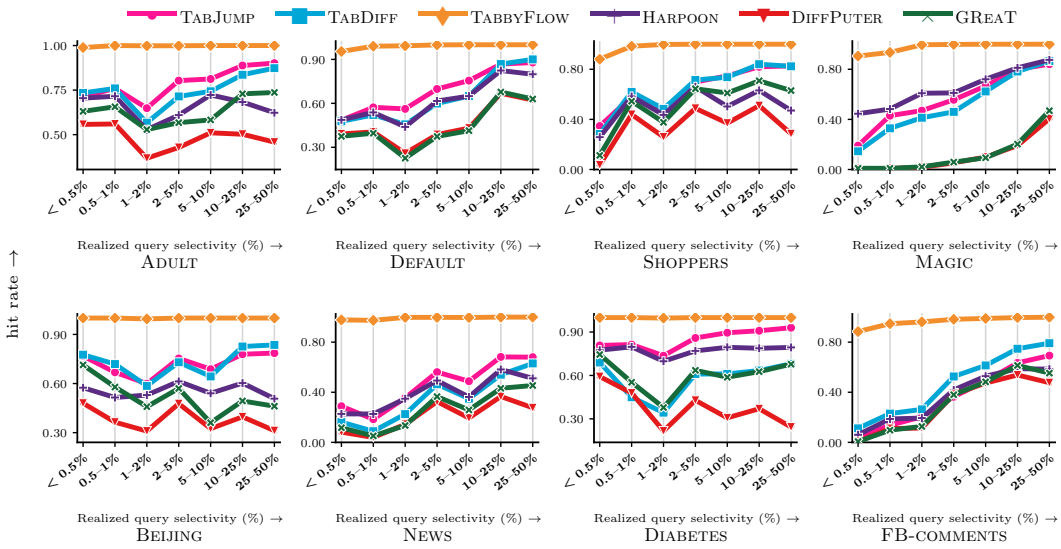


Figure 3: Conditional-generation hit rate across realized query selectivity on all eight datasets. Each panel uses its own vertical scale.

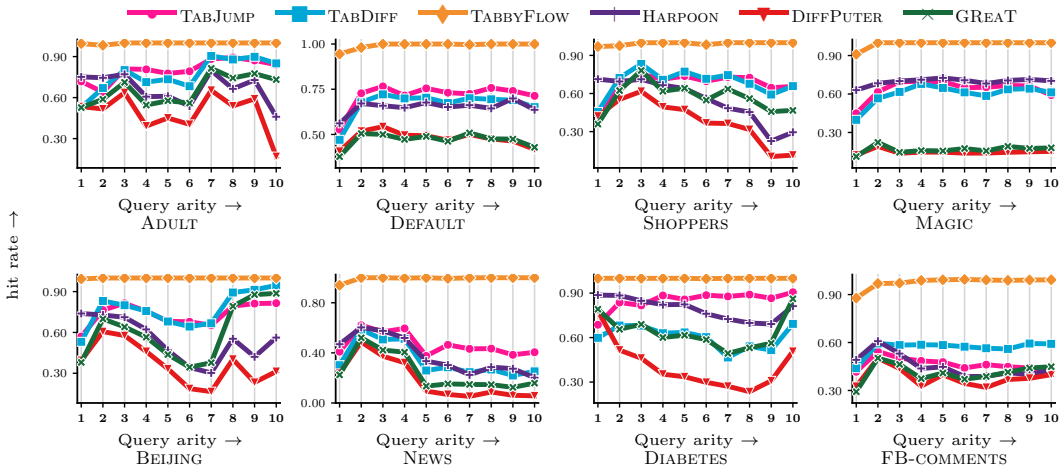


Figure 4: Conditional-generation hit rate across query arity on all eight datasets. Each panel uses its own vertical scale.

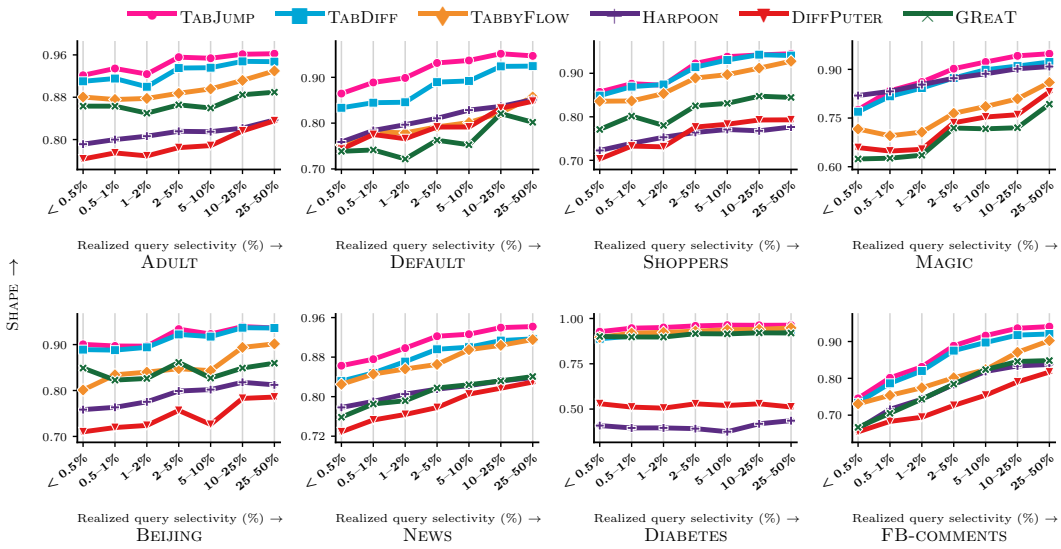


Figure 5: Conditional-generation column-shape score across realized query selectivity on all eight datasets. Each panel uses its own vertical scale.

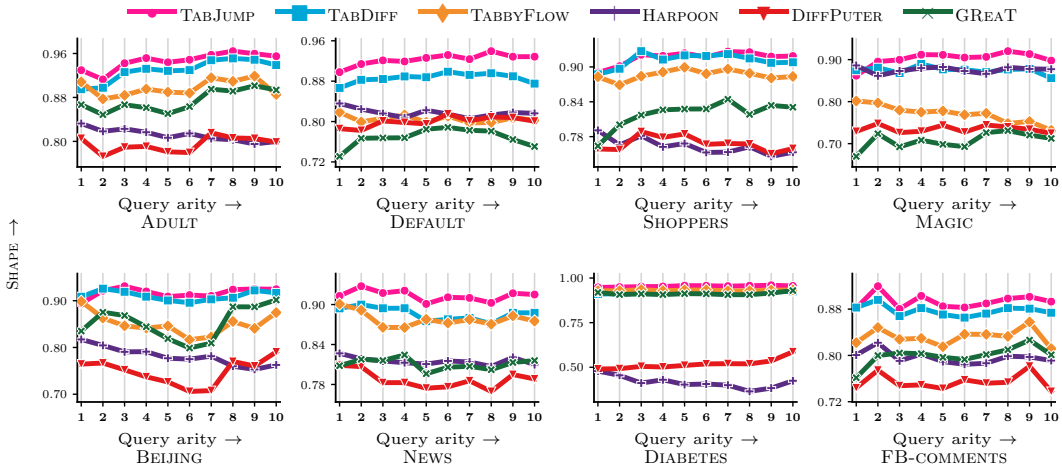


Figure 6: Conditional-generation column-shape score across query arity on all eight datasets. Each panel uses its own vertical scale.

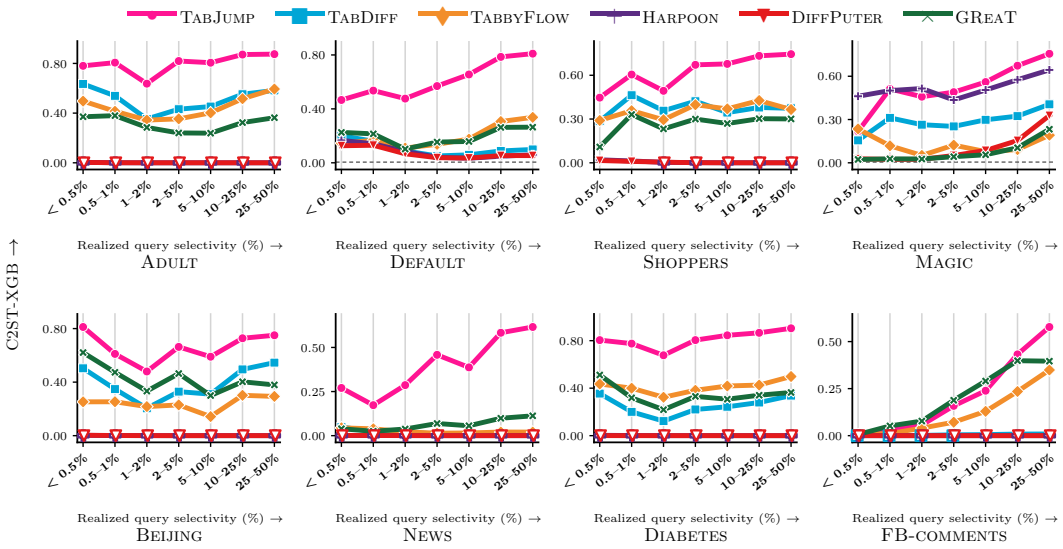


Figure 7: Conditional-generation C2ST-XGBoost similarity across realized query selectivity on all eight datasets. Each panel uses its own vertical scale.

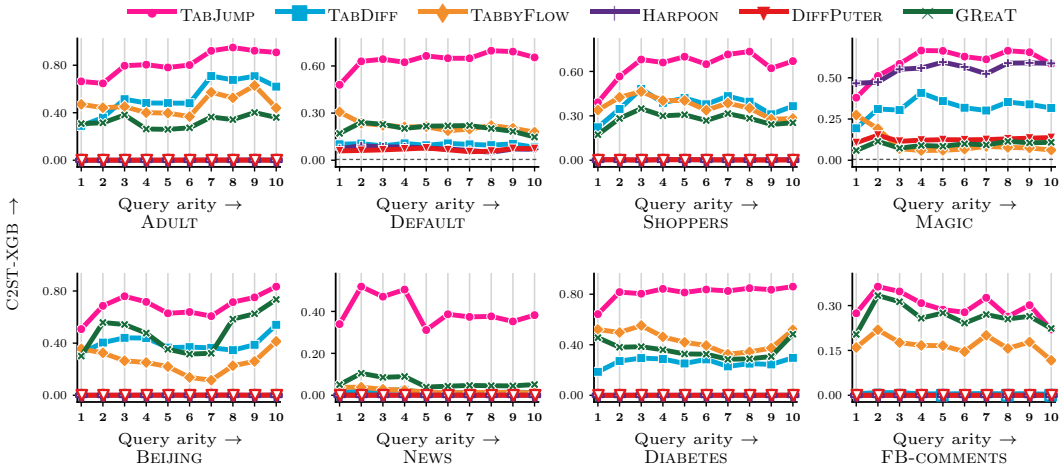


Figure 8: Conditional-generation C2ST-XGBoost similarity across query arity on all eight datasets. Each panel uses its own vertical scale.