# ONE-STEP GRAPH SAMPLING BY BELIEF FLOW MAPS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Diffusion-like graph generation typically relies on iterative sampling, where the generated state is progressively refined toward the data distribution. For graph generation, we adopt a categorical Bayesian flow that naturally accommodates graph structures. Our key insight is to derive a particularly simple probability flow ODE that linearly links the belief dynamics to data prediction. This ODE provides a principled foundation that allows one-step sampling and facilitates trajectory-based distillation. The resulting continuous trajectory allows one-step generation to be formulated as learning a finite-time transition along the Bayesian flow. Distilling graph generation into a single step, however, is challenging. We argue that effective one-step graph generation should preserve not only the endpoints of the sampling process, but also its underlying trajectory. Based on this observation, we propose Belief Flow Maps (BFM), a trajectory-aware distillation framework. In BFM, belief chord distillation performs local transitions along the probability flow trajectory and distills their refined endpoint into a single large transition, while belief tangent distillation provides complementary local supervision by aligning the learned transition with the belief flow velocity at the reached belief state. Together, they capture both long-range transport and local trajectory dynamics. Experiments on molecular and structural graph generation benchmarks demonstrate that BFM substantially improves one-step generation quality over existing methods.

## 1 INTRODUCTION

Graph generation is a fundamental problem in structured generative modeling, with applications ranging from molecular design and materials discovery to network synthesis and program generation (Gómez-Bombarelli et al., 2018; Du et al., 2024; Schneuing et al., 2024; Guo & Schwaller, 2026). Unlike image generation, where the output lies in a dense continuous space, graphs contain discrete node and edge variables whose configurations are strongly coupled. Recent diffusion- and flow-based approaches address this challenge by progressively refining graph states through multiple sampling steps (Vignac et al., 2023; Eijkelboom et al., 2024; Qin et al., 2025; Petersen et al., 2026). Such iterative refinement improves generation quality, but requires repeated evaluations of the generative model, making sampling a major computational bottleneck.

A natural way to reduce this cost is to distill the multi-step generative process into a small number of finite-time transitions. Recent flow map distillation methods have demonstrated the effectiveness of this idea for continuous generative models (Boffi et al., 2025). However, dedicated one-step sampling for graph generation remains largely unexplored. This gap is nontrivial: the structural refinement of coupled discrete node and edge variables can be distributed across the sampling trajectory. Simply replacing a sequence of local updates with a single endpoint-to-endpoint prediction may therefore lose the structural refinement captured by the original sampling process. This raises a natural question: can the entire generative trajectory be accurately approximated with only one or a few model evaluations?

One-step generation can be viewed as learning a finite-time transport map that replaces many local updates with a single large transition. Such a view is closely related to flow map distillation, where a model learns to transport an initial state to a later point along a continuous generative trajectory (Boffi et al., 2025). However, directly distilling the endpoints of a long transition is particularly challenging for graph generation. A large transition must simultaneously resolve many coupled discrete decisions that are progressively refined during multi-step sampling. Supervising only the final graph therefore provides little information about how the belief should evolve between the two endpoints, potentially causing the learned transition to deviate from the original generative trajectory. This suggests that



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.

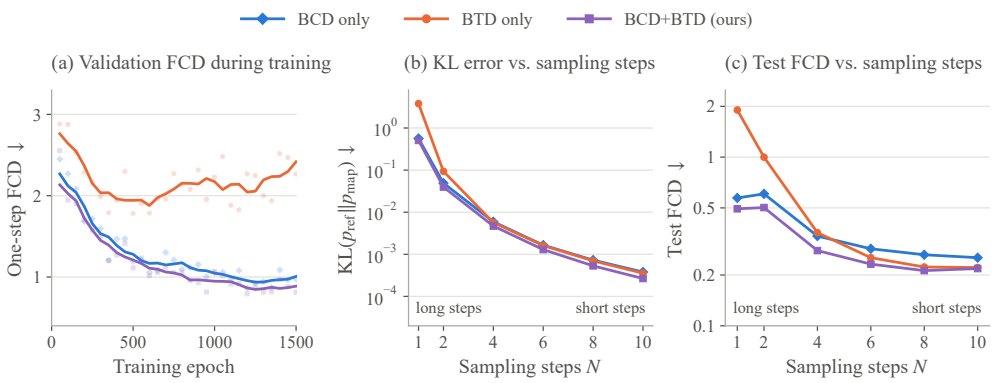


Figure 1: Accurate few-step generation requires full-interval supervision. All panels use QM9; **(b,c)** are evaluated at epoch 1500. **(a)** One-step validation FCD (1k molecules) during training. BCD provides full-interval supervision that reduces one-step FCD; adding BTD further lowers the final FCD. **(b)** $\mathrm{KL}(p_{\text{ref}}\|p_{\text{map}})$ (1k molecules) versus sampling steps $N$. $p_{\text{map}}$ is the prediction from the learned map, and $p_{\text{ref}}$ is the reference from 100-step sampling. BCD reduces long-step error, and adding BTD further reduces it at every tested $N$. **(c)** Test FCD (10k molecules) versus $N \in \{1, 2, 4, 6, 8, 10\}$. Adding BTD to BCD improves FCD at every sampling budget.

effective one-step distillation should preserve not only long-range endpoint transport, but also the local dynamics of the underlying trajectory.

To make this question tractable, we revisit the continuous dynamics underlying categorical Bayesian Sample Inference (BSI) (Lienen et al., 2026; Petersen et al., 2026). BSI represents discrete graph variables through continuous categorical beliefs and progressively refines these beliefs toward a discrete graph, providing a natural continuous space in which the evolution of graph structure can be modeled. However, its probability flow dynamics do not expose a simple form that explicitly characterizes the velocity of the belief state. To address this, we reformulate the BSI dynamics through an appropriate transformation of the belief state and time, yielding a particularly simple ODE whose belief flow velocity is directly given by data prediction (§2.2). This reformulation exposes an explicit continuous trajectory in the transformed belief space, along which one-step generation can be viewed as learning a finite-time transition rather than directly predicting the final graph. Moreover, integrating the probability flow ODE over several local steps naturally provides refined targets for this finite-time transition. This view turns one-step graph generation from direct endpoint prediction into trajectory-aware transport learning, providing the foundation for our distillation framework.

Based on this ODE, we propose Belief Flow Maps (BFM), a trajectory-aware distillation framework for efficient graph generation. BFM combines two complementary forms of supervision. Belief Chord Distillation (BCD) integrates the probability flow trajectory over several local steps and uses the resulting endpoint to supervise a single large transition, preserving long-range transport. Belief Tangent Distillation (BTD) provides local supervision by aligning the learned transition with the instantaneous belief flow velocity at the reached belief state, encouraging consistency with the underlying trajectory (Song et al., 2023; Boffi et al., 2025; Roos et al., 2026). As illustrated in Figure 1, BCD constrains the full-interval transport, while BTD further refines the transition through a local constraint at the endpoint. Together, they capture complementary aspects of the belief trajectory.

Our main contributions can be summarized as follows:

- We derive a simple probability flow ODE from BSI through an appropriate transformation of the belief state and time, providing a continuous trajectory formulation for efficient graph generation.
- We propose BFM, a trajectory-aware distillation framework that combines long-range transport supervision through BCD with local trajectory supervision through BTD.
- We demonstrate that BFM substantially improves one-step graph generation across molecular and structural benchmarks, achieving the best distributional fidelity among compared one-step methods.

For efficient decoding, we further introduce a lightweight terminal head that converts the final belief into a discrete graph without requiring an additional backbone evaluation. For molecular graphs, a valence-aware penalty is incorporated to improve chemical validity. Experiments on multiple graph generation benchmarks show that BFM substantially improves one-step and two-step generation quality over existing methods.

## 2 METHOD

### 2.1 BELIEF UPDATES

Consider a graph with $n$ nodes, where each node and candidate edge is associated with a categorical label. We concatenate all one-hot labels into a graph data vector $\boldsymbol{x} = [x_1; \ldots; x_K]$ where each $x_k$ denotes one categorical variable. For each categorical component $x_k$ with $K$ possible classes, we parameterize its belief distribution by a logit vector $\boldsymbol{z} = [z_1; \ldots; z_K]$, such that $p(x_k \mid \boldsymbol{z}) = e^{z_k} / \sum_{c=1}^{K} e^{z_c}$. Given the current belief state $\boldsymbol{z}_i$ at time $t_i$, Bayesian Sample Inference (BSI) (Lienen et al., 2026; Petersen et al., 2026) performs a Bayesian update from $t_i$ to $t_{i+1}$. Define $\alpha_i = \beta_{t_{i+1}} - \beta_{t_i}$. A noisy observation of the underlying data $\boldsymbol{x}$ is then sampled as

$$\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{x}, \alpha_i^{-1} \boldsymbol{I}) \,. \tag{1}$$

The belief logits are updated as

$$\boldsymbol{z}_{i+1} \leftarrow \boldsymbol{z}_i + \alpha_i \boldsymbol{y}_i \,. \tag{2}$$

Thus, BSI generates a sample by repeatedly refining the belief state through a sequence of Bayesian updates: $\boldsymbol{z}_0 \to \boldsymbol{z}_1 \to \cdots \to \boldsymbol{z}_T$ . The belief logit $\boldsymbol{z}_T$ is converted into categorical probabilities by applying softmax, from which the one-hot labels can be sampled.

If we use Eq. 2 to generate data from the prior $\boldsymbol{z}_0$, we need a network $\boldsymbol{x}^\theta$ to learn $\boldsymbol{x}$ in each step. The training objective of BSI follows directly from the marginal form of the Bayesian update. Rather than explicitly simulating the full sequence of intermediate updates, BSI constructs the belief state at an arbitrary time $t_i$ in a single step. Specifically, consider the accumulated update from $t_0$ to $t_i$ from Eq. 2, and define $\alpha = \sum_{j=0}^{i-1} \alpha_j = \beta_{t_i} - \beta_{t_0}$. By aggregating the intermediate Bayesian observations over this interval, the resulting observation can be equivalently expressed as

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{x}, \alpha^{-1}\mathbf{I}), \tag{3}$$

which induces the corresponding belief state

$$\boldsymbol{z}_i = \boldsymbol{z}_0 + \alpha \boldsymbol{y}. \tag{4}$$

This allows direct sampling from the marginal belief state at any time $t_i$. We initialize $\boldsymbol{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$, where $\sigma_0 > 0$ is the prior standard deviation. Given $\boldsymbol{z}_i$ and its associated time $t_i$, the neural network is trained to reconstruct the underlying data $\boldsymbol{x}$: $\boldsymbol{x}^\theta(\boldsymbol{z}_i, t_i)$. Following BSI (Lienen et al., 2026; Petersen et al., 2026), the resulting training objective is given by the reconstruction loss $\mathcal{L}_{\mathrm{BF}} = \frac{\beta'_{t_i}}{2} \|\boldsymbol{x}^\theta(\boldsymbol{z}_i, t_i) - \boldsymbol{x}\|^2$. Consequently, training directly samples from the belief marginal at arbitrary time $t_i$ and learns to reconstruct $\boldsymbol{x}$.

### 2.2 BELIEF FLOW

**From Belief Updates to Continuous Dynamics.** The stochastic dynamics above induce a time-dependent probability density $q_t(\boldsymbol{z}|\boldsymbol{x})$ over the belief state. To characterize its evolution, we generalize the belief flow in the form of a stochastic differential equation (SDE).

Using Eq. 1, we write $\boldsymbol{y}_i = \boldsymbol{x} + \alpha_i^{-1/2} \boldsymbol{\epsilon}_i$, where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Substituting into Eq. 2 gives

$$\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + \alpha_i \boldsymbol{x} + \sqrt{\alpha_i} \boldsymbol{\epsilon}_i. \tag{5}$$

This decomposition separates the update into a deterministic component, which shifts the belief state toward the underlying data $\boldsymbol{x}$, and a stochastic component induced by the observation uncertainty.

The increment $\alpha_i$ satisfies $\alpha_i = \beta_{t_{i+1}} - \beta_{t_i} \approx \beta'_t \Delta t$. Accordingly, the discrete update converges to the following continuous-time stochastic differential equation:

$$\mathrm{d}\boldsymbol{z}_t = \beta'_t \boldsymbol{x} \mathrm{d}t + \sqrt{\beta'_t} \mathrm{d}\boldsymbol{w}_t. \tag{6}$$

**From the SDE to the Probability Flow ODE.** Let $q_t(\boldsymbol{z}_t|\boldsymbol{x})$ denote the conditional density induced by Eq. 6 for a given sample $\boldsymbol{x}$. The corresponding Fokker-Planck equation is

$$\frac{\partial q_t(\boldsymbol{z}_t|\boldsymbol{x})}{\partial t} = -\nabla_{\boldsymbol{z}_t} \cdot (\beta_t' \boldsymbol{x}\, q_t(\boldsymbol{z}_t|\boldsymbol{x})) + \frac{\beta_t'}{2} \Delta q_t(\boldsymbol{z}_t|\boldsymbol{x}). \tag{7}$$

For a given $\boldsymbol{x}$, its drift in Eq. 6 is $\beta_t'\boldsymbol{x}$. Since the marginal density $q_t(\boldsymbol{z}_t)$ is obtained by marginalizing over the data distribution: $q_t(\boldsymbol{z}_t) = \int q_t(\boldsymbol{z}_t|\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$, integrating Eq. 7 with respect to $p(\boldsymbol{x})$ gives

$$\frac{\partial q_t(\boldsymbol{z}_t)}{\partial t} = -\nabla_{\boldsymbol{z}_t} \cdot \left[ \left( \beta_t' \mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t] - \frac{\beta_t'}{2} \nabla_{\boldsymbol{z}_t} \log q_t(\boldsymbol{z}_t) \right) q_t(\boldsymbol{z}_t) \right]. \tag{8}$$

This expression takes the form of a continuity equation and identifies a deterministic field whose induced density evolution matches that of the stochastic process. Consequently, the probability flow ODE can be represented by

$$\frac{\mathrm{d}\boldsymbol{z}_t}{\mathrm{d}t} = \beta_t' \mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t] - \frac{\beta_t'}{2} \nabla_{\boldsymbol{z}_t} \log q_t(\boldsymbol{z}_t). \tag{9}$$

We next express both terms in Eq. 9 using the same conditional expectation $\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t]$.

By Eq. 4 and $\boldsymbol{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$, the conditional belief is Gaussian, $q_t(\boldsymbol{z}_t|\boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{\mu}_0 + \beta_t \boldsymbol{x},\ \sigma_t^2 I\big)$ where we define $\sigma_t^2 = \sigma_0^2 + \beta_t$. Its conditional score is $\nabla_{\boldsymbol{z}_t} \log q_t(\boldsymbol{z}_t|\boldsymbol{x}) = -(\boldsymbol{z}_t - \boldsymbol{\mu}_0 - \beta_t \boldsymbol{x})/\sigma_t^2$. Using the conditional-to-marginal score identity, we obtain the marginal score in Eq. 9 by

$$\nabla_{\boldsymbol{z}_t} \log q_t(\boldsymbol{z}_t) = \mathbb{E}_{q_t(\boldsymbol{x}|\boldsymbol{z}_t)} \big[ \nabla_{\boldsymbol{z}_t} \log q_t(\boldsymbol{z}_t|\boldsymbol{x}) \big] = -\frac{\boldsymbol{z}_t - \boldsymbol{\mu}_0 - \beta_t \mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t]}{\sigma_t^2} \tag{10}$$

The same conditional expectation, $\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t]$, determines both terms in Eq. 9. Under the squared-error loss, its optimal estimator, $\boldsymbol{x}^\theta$, is the conditional mean $\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t]$ (Song et al., 2021). We use $\boldsymbol{x}^\theta$ to approximate this quantity: $\boldsymbol{x}^\theta \approx \mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t]$. Substituting this single approximation into both terms of Eq. 9 and Eq. 10 yields

$$\frac{\mathrm{d}\boldsymbol{z}_t}{\mathrm{d}t} = \beta_t' \Big( 1 - \frac{\beta_t}{2\sigma_t^2} \Big) \boldsymbol{x}^\theta + \frac{\beta_t'}{2\sigma_t^2} (\boldsymbol{z}_t - \boldsymbol{\mu}_0). \tag{11}$$

To expose the underlying linear structure, we introduce the normalized belief coordinate and the corresponding time reparameterization:

$$\widetilde{\boldsymbol{z}}_t = \frac{\boldsymbol{z}_t - \boldsymbol{\mu}_0}{\sigma_t}, \qquad \sigma_t = \sqrt{\sigma_0^2 + \beta_t}, \qquad \tau_t = \frac{\beta_t}{\sigma_t}. \tag{12}$$

To enable efficient graph generation with the categorical Bayesian belief flow, we derive a belief flow ODE that explicitly characterizes the velocity of the belief state. Substituting Eq. 12 into Eq. 11 (Appendix A.1) yields the simple form of belief flow ODE,

$$\frac{\mathrm{d}\widetilde{\boldsymbol{z}}_t}{\mathrm{d}\tau_t} = \boldsymbol{x}^\theta. \tag{13}$$

This reparameterization is important for our distillation framework. In the normalized belief space, the belief flow velocity is given directly by the predicted data $\boldsymbol{x}^\theta$, eliminating the explicit dependence on the current belief state in Eq. 11. Consequently, graph generation can be viewed as transporting the normalized belief along a continuous trajectory driven by data prediction, and a finite-time transition can be obtained by integrating $\boldsymbol{x}^\theta$ along this trajectory. This formulation, Eq. 13, provides a natural trajectory on which finite-time transitions can be defined, enabling us to construct both long-range and local supervision for distillation.

**One-step belief transport.** Given the normalized belief ODE, we can replace the many small steps along the curved trajectory with a single straight step, reducing sampling to one or a few evaluations. Integrating from $s$ to $t$ gives $\widetilde{\boldsymbol{z}}_t - \widetilde{\boldsymbol{z}}_s = (\tau_t - \tau_s)\,\overline{\boldsymbol{x}}_{s,t}$, where $\overline{\boldsymbol{x}}_{s,t}$ is this rate of change averaged over the interval. We learn a predictor $\boldsymbol{x}^\theta_{s,t}(\boldsymbol{z}_s)$ that estimates the slope of the straight line from $\widetilde{\boldsymbol{z}}_s$ to $\widetilde{\boldsymbol{z}}_t$, and abbreviate it $\boldsymbol{x}^\theta_{s,t}$. When $s = t$ the line collapses to a point and the slope becomes the tangent at $\widetilde{\boldsymbol{z}}_t$, recovering the local reconstructor: $\boldsymbol{x}^\theta_{t,t} = \boldsymbol{x}^\theta(\boldsymbol{z}_t, t)$. Parameterization details are in Appendix A.2, *Predictor parameterization*. Substituting the learned slope $\boldsymbol{x}^\theta_{s,t}$ for the true average $\overline{\boldsymbol{x}}_{s,t}$ gives the one-step approximation $\widetilde{\boldsymbol{z}}^\theta_{s,t}$ to the true target $\widetilde{\boldsymbol{z}}_t$:

$$\widetilde{\boldsymbol{z}}^\theta_{s,t} = \widetilde{\boldsymbol{z}}_s + (\tau_t - \tau_s)\boldsymbol{x}^\theta_{s,t}. \tag{14}$$

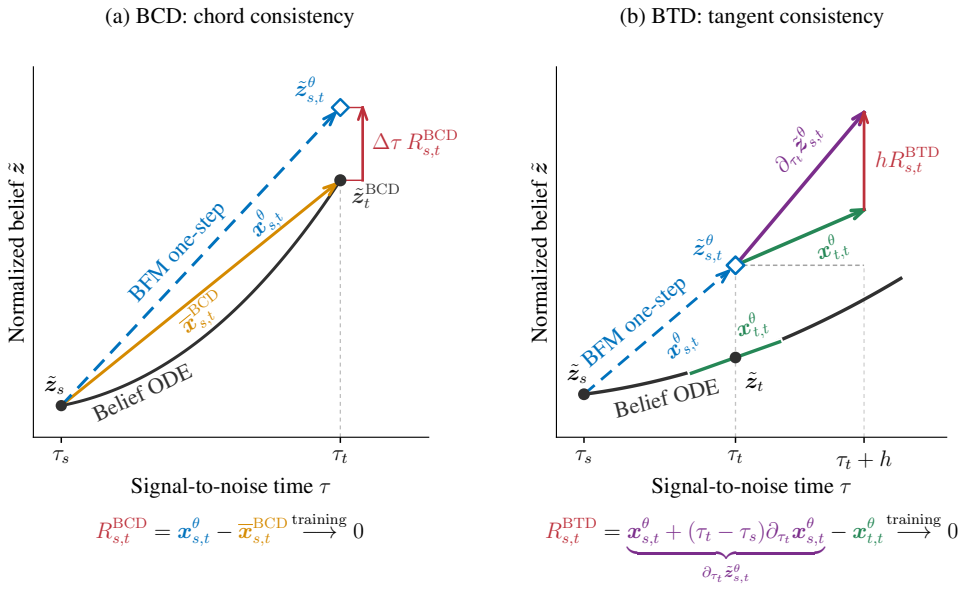


Figure 2: BCD and BTD supervise the belief flow map at complementary scales along the ODE trajectory. **(a)** BCD is a *chord* constraint: it matches the map's interval-average prediction $\boldsymbol{x}^{\theta}_{s,t}$ (⇢) to the ODE-integrated average $\overline{\boldsymbol{x}}^{\mathrm{BCD}}_{s,t}$ (→), supervising the full displacement from $\tilde{\boldsymbol{z}}_s$ to $\tilde{\boldsymbol{z}}^{\theta}_{s,t}$. The chord residual $R^{\mathrm{BCD}}_{s,t}$ is shown as ↑. **(b)** BTD is a *tangent* constraint: it aligns the map derivative $\partial_{\tau_t}\tilde{\boldsymbol{z}}^{\theta}_{s,t}$ (→) with the local tangent slope $\boldsymbol{x}^{\theta}_{t,t}$ (→) at the predicted endpoint $\tilde{\boldsymbol{z}}^{\theta}_{s,t}$ (◇). This ensures local accuracy at $\tau_t$ and complements the displacement supervision of BCD. The residual $R^{\mathrm{BTD}}_{s,t}$ is shown as ↑. BCD anchors long-step transport through full-interval supervision, and BTD further refines each transition, together yielding more accurate results than either alone, as shown in Figure 1b.

### 2.3 Belief Chord Distillation

The map (Eq. 14) is exact when its slope equals the average rate of change of the belief over the interval. Instead of directly distilling the endpoint of a large transition, Belief Chord Distillation (BCD) obtains this average from the trajectory itself, providing a global constraint on the map's total displacement from $\tilde{\boldsymbol{z}}_s$ to $\tilde{\boldsymbol{z}}^{\theta}_{s,t}$: it first performs several local steps along the normalized belief ODE, which progressively refine the graph structure, and then uses the resulting trajectory endpoint as the distillation target for a single large transition. Formally, integrating Eq. 13 from $s$ to $t$ gives

$$\tilde{\boldsymbol{z}}_t - \tilde{\boldsymbol{z}}_s = (\tau_t - \tau_s)\overline{\boldsymbol{x}}_{s,t}, \qquad \overline{\boldsymbol{x}}_{s,t} = \frac{1}{\tau_t - \tau_s}\int_s^t \boldsymbol{x}^{\theta}_{u,u}\,\tau'_u\,\mathrm{d}u. \tag{15}$$

The map (Eq. 14) recovers the true $\tilde{\boldsymbol{z}}_t$ when the learned slope $\boldsymbol{x}^{\theta}_{s,t}$ equals the ODE average. We approximate this average by stopped-gradient Heun integration of Eq. 13 on a grid $s = t_0 < \cdots < t_d = t$ with step sizes $\delta_j = \tau_{t_{j+1}} - \tau_{t_j}$. Writing $v_j(\tilde{\boldsymbol{z}}) = \boldsymbol{x}^{\theta}(\boldsymbol{\mu}_0 + \sigma_{t_j}\tilde{\boldsymbol{z}}, t_j)$ for the belief velocity at $t_j$ and starting from $\tilde{\boldsymbol{z}}^{(0)} = \tilde{\boldsymbol{z}}_s$, each Heun substep averages the velocities at the start and predicted end of the interval, $\bar{v}_j = \frac{1}{2}\big[v_j(\tilde{\boldsymbol{z}}^{(j)}) + v_{j+1}(\tilde{\boldsymbol{z}}^{(j)} + \delta_j v_j(\tilde{\boldsymbol{z}}^{(j)}))\big]$, and advances the belief by $\tilde{\boldsymbol{z}}^{(j+1)} = \tilde{\boldsymbol{z}}^{(j)} + \delta_j\,\bar{v}_j$. The BCD target is the average slope of the integrated trajectory:

$$\overline{\boldsymbol{x}}^{\mathrm{BCD}}_{s,t} = \frac{1}{\tau_t - \tau_s}\sum_{j=0}^{d-1}\delta_j\,\bar{v}_j. \tag{16}$$

Algorithm 3 in Appendix A.3 summarizes these steps. In Figure 2a, the map's prediction uses $\boldsymbol{x}^{\theta}_{s,t}$ (⇢), the ODE-integrated average is $\overline{\boldsymbol{x}}^{\mathrm{BCD}}_{s,t}$ (→), and their mismatch is $R^{\mathrm{BCD}}_{s,t}$. The BCD loss minimizes this gap:

$$\mathcal{L}_{\mathrm{BCD}} = \mathbb{E}\left[\left\|\boldsymbol{x}^{\theta}_{s,t} - \mathrm{sg}\big[\overline{\boldsymbol{x}}^{\mathrm{BCD}}_{s,t}\big]\right\|^2_G\right]. \tag{17}$$

Because the target is integrated over the full interval, BCD directly supervises the total displacement $\widetilde{\boldsymbol{z}}^{\theta}_{s,t} - \widetilde{\boldsymbol{z}}_s$ along the chord, so the local multi-step trajectory provides informative supervision while the map keeps the cost of a single evaluation.

### 2.4 Belief Tangent Distillation

BCD globally constrains the full-interval displacement but does not directly regulate the map's local behavior, and on its own it does not compose well across shorter intervals (§3.3). Belief Tangent Distillation (BTD) adds pointwise supervision at the predicted endpoint: it requires the map derivative there to match the instantaneous ODE belief flow velocity. When the map is exact, $\widetilde{\boldsymbol{z}}^{\theta}_{s,t} = \widetilde{\boldsymbol{z}}_t$, and differentiating by $\tau_t$ gives $\partial_{\tau_t}\widetilde{\boldsymbol{z}}^{\theta}_{s,t} = \boldsymbol{x}^{\theta}_{t,t} = \boldsymbol{x}^{\theta}(\boldsymbol{z}_t, t)$ by the normalized belief ODE (Eq. 13). The intuition is that the slope of the line from $\widetilde{\boldsymbol{z}}_s$ to $\widetilde{\boldsymbol{z}}_t$ must converge to the tangent slope at $\widetilde{\boldsymbol{z}}_t$ as the interval shrinks, as illustrated in Figure 2b, where the map derivative $\partial_{\tau_t}\widetilde{\boldsymbol{z}}^{\theta}_{s,t}$ ($\rightarrow$) approaches the local tangent slope $\boldsymbol{x}^{\theta}_{t,t}$ ($\rightarrow$). BTD enforces this for the learned map (Song et al., 2023; Boffi et al., 2025; Roos et al., 2026): $\partial_{\tau_t}\widetilde{\boldsymbol{z}}^{\theta}_{s,t} \approx \boldsymbol{x}^{\theta}_{t,t}$. Expanding the map derivative gives the residual

$$R^{\mathrm{BTD}}_{s,t} = \partial_{\tau_t}\widetilde{\boldsymbol{z}}^{\theta}_{s,t} - \mathrm{sg}\big[\boldsymbol{x}^{\theta}_{t,t}\big] = \boldsymbol{x}^{\theta}_{s,t} + (\tau_t - \tau_s)\,\partial_{\tau_t}\boldsymbol{x}^{\theta}_{s,t} - \mathrm{sg}\big[\boldsymbol{x}^{\theta}(\boldsymbol{z}^{\theta}_{s,t}, t)\big], \tag{18}$$

and we minimize

$$\mathcal{L}_{\mathrm{BTD}} = \mathbb{E}\left[\left\|R^{\mathrm{BTD}}_{s,t}\right\|^2_G\right]. \tag{19}$$

sg stops gradients through the endpoint target $\boldsymbol{x}^{\theta}_{t,t}$. The constraint acts on the map derivative, while the predictor $\boldsymbol{x}^{\theta}_{s,t}$ represents an interval average. BCD constrains the total displacement over the full interval (chord), and BTD further regulates the instantaneous behavior at the predicted endpoint (tangent), together supervising the map at complementary scales.

Beyond these two distillation losses, the base objective includes the reconstruction loss $\mathcal{L}_{\mathrm{BF}}$ (§2.1), a terminal-head distillation loss $\mathcal{L}_{\mathrm{head}}$ that trains a lightweight decoder to map the final belief to a discrete graph without a second backbone evaluation, and a valence penalty $\mathcal{L}_{\mathrm{val}}$ that encourages chemical consistency for molecular graphs (details in Appendix A.4 and A.5): $\mathcal{L}_{\mathrm{base}} = \mathcal{L}_{\mathrm{BF}} + \mathcal{L}_{\mathrm{head}} + \lambda_{\mathrm{val}}\mathcal{L}_{\mathrm{val}}$. The full training loss adds the two distillation terms with nonnegative weights:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{base}} + \lambda_{\mathrm{BCD}}\mathcal{L}_{\mathrm{BCD}} + \lambda_{\mathrm{BTD}}\mathcal{L}_{\mathrm{BTD}}. \tag{20}$$

Appendix A.3 summarizes training in Algorithm 1 and the BCD target in Algorithm 3.

### 2.5 Few-Step Graph Generation

We first sample the node count from its empirical training distribution, then initialize and transport the corresponding node and edge beliefs. Training produces the predictor $\boldsymbol{x}^{\theta}_{s,t}$, which estimates the slope of the belief trajectory over any interval $[s, t]$. At sampling time, we convert this into a logit space update by substituting $\boldsymbol{x}^{\theta}_{s,t}$ into the normalized coordinate map (Eq. 14) and transforming back to the original coordinates (derivation in Appendix A.2):

$$\boldsymbol{z}^{\theta}_{s,t}(\boldsymbol{z}_s) = \boldsymbol{\mu}_0 + a_{s,t}(\boldsymbol{z}_s - \boldsymbol{\mu}_0) + (\beta_t - a_{s,t}\beta_s)\,\boldsymbol{x}^{\theta}_{s,t}, \tag{21}$$

where $a_{s,t} = \sigma_t/\sigma_s$ rescales the source belief to the destination noise scale. For $K$ transport steps on an inference grid $0 = t_0 < \cdots < t_K = 1$, we sample $\boldsymbol{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$ and apply

$$\boldsymbol{z}_{t_{k+1}} = \boldsymbol{z}^{\theta}_{t_k,t_{k+1}}(\boldsymbol{z}_{t_k}), \qquad k = 0, \ldots, K-1. \tag{22}$$

With few transport steps, the final belief $\boldsymbol{z}_{t_K}$ may still spread across multiple categories rather than collapsing onto one. A learned terminal decoder $g_\psi$ sharpens this belief into per-block category distributions, and a blockwise argmax selects the discrete graph $G$:

$$G = \mathrm{decode}(\boldsymbol{z}_{t_K}) = \arg\max g_\psi(\boldsymbol{z}_{t_K}), \tag{23}$$

where the argmax is applied independently to each node and edge block (parameterization details are in Appendix A.4). Algorithm 2 in Appendix A.3 summarizes the full procedure.

Table 1: Molecular generation on QM9 and ZINC250k. Multi-step baselines are from Roos et al. (2026) except DeFoG (500), GraphBSI, and CFM, which we evaluate. **Bold** marks the best value.

| | | QM9 | | | ZINC250k | | |
|---|---|---|---|---|---|---|---|
| **Method** | **NFE** | Valid ↑ | V.U. ↑ | FCD ↓ | Valid ↑ | V.U. ↑ | FCD ↓ |
| *Multi-step references* | | | | | | | |
| GDSS | 1000 | 95.7 | 94.3 | 2.90 | 97.0 | 96.6 | 14.7 |
| GruM | 1000 | 99.7 | – | 0.11 | 98.7 | – | 2.26 |
| CatFlow | 100 | 99.8 | 99.7 | 0.44 | 99.2 | 99.2 | 13.2 |
| DeFoG | 50 | 99.2 | 95.4 | 0.26 | 96.7 | 96.6 | 2.12 |
| DeFoG | 500 | 99.0 | 95.3 | 0.15 | 94.1 | 94.0 | 1.42 |
| GraphBSI | 10 | 97.7 | 93.5 | 0.28 | 96.4 | 96.4 | 7.83 |
| GraphBSI | 50 | 98.9 | 96.7 | 0.16 | 97.8 | 97.8 | 5.44 |
| *One-step generation* (NFE $= 1$) | | | | | | | |
| MoFlow | 1 | 91.4 | 90.2 | 4.46 | 63.1 | 63.0 | 20.9 |
| Naive Flow Map | 1 | 84.6 | 78.7 | 2.31 | 59.5 | 59.4 | 21.7 |
| GraphBSI | 1 | **98.4** | 6.3 | 8.59 | **100.0** | 0.3 | 45.5 |
| CFM | 1 | 95.1 | 83.6 | 2.27 | 91.1 | 90.8 | 20.1 |
| BFM (ours) | 1 | 96.4 | **93.3** | **0.39** | 91.3 | **91.3** | **14.6** |
| *Two-step generation* (NFE $= 2$) | | | | | | | |
| Naive Flow Map | 2 | 87.7 | 85.7 | 0.69 | 51.6 | 51.6 | 16.1 |
| GraphBSI | 2 | 83.2 | 72.8 | 2.07 | **94.4** | 52.9 | 28.1 |
| CFM | 2 | 96.3 | 93.2 | 0.61 | 80.3 | 80.3 | 12.8 |
| BFM (ours) | 2 | **97.1** | **94.0** | **0.34** | 92.0 | **92.0** | **11.2** |

# 3 Experiments

## 3.1 Experimental Setup

**Benchmarks**. We evaluate molecular generation on QM9 (Ramakrishnan et al., 2014), ZINC250k (Sterling & Irwin, 2015), and MOSES (Polykovskiy et al., 2020), covering small to drug-like molecules. We evaluate structural graph generation on Tree (Bergmeister et al., 2024; Qin et al., 2025), where validity requires connectivity and acyclicity. Dataset details are in Appendix A.7.

**Baselines**. On QM9 and ZINC250k, we compare with MoFlow (Zang & Wang, 2020), Naive Flow Maps (Roos et al., 2026), Categorical Flow Maps (Roos et al., 2026), and GraphBSI (Petersen et al., 2026) at low NFE, and with GDSS (Jo et al., 2022), GruM (Jo et al., 2024), CatFlow (Eijkelboom et al., 2024), DeFoG (Qin et al., 2025), and GraphBSI Petersen et al. (2026) at larger budgets. For MOSES and Tree, we include discrete diffusion and flow models from the DeFoG benchmark protocol (Qin et al., 2025), and for Tree we include the baseline HSpectre (Bergmeister et al., 2024).

**Metrics**. NFE counts backbone evaluations during sampling. For QM9 and ZINC250k, we report validity (Valid), Fréchet ChemNet Distance (FCD), and V.U., the product of Valid and uniqueness (Unique). For MOSES, we report Valid, V.U., V.U.N., the product of V.U. and novelty (Novel), filter pass rate (Filters), FCD, nearest-neighbor similarity (SNN), and scaffold similarity (Scaf.) (Polykovskiy et al., 2020; Qin et al., 2025). For Tree, we report Valid, Unique, Novel, V.U.N., and the maximum mean discrepancy (MMD) of degree (Deg.), clustering coefficient (Clus.), orbit counts (Orbit), spectral (Spec.), and wavelet (Wavelet) features. Ratio averages these MMDs, each normalized by the train-test MMD (Martinkus et al., 2022; Qin et al., 2025).

## 3.2 Main Comparison

**QM9 and ZINC250k**. BFM achieves the lowest FCD and highest V.U. among the compared methods at each of the one- and two-step budgets on both datasets, as shown in Table 1. On ZINC250k, other methods sacrifice validity when adding a second step, whereas BFM improves both validity and FCD. On QM9, BFM likewise improves both and maintains the best V.U. and FCD at each budget.

Table 2: Molecular generation on MOSES. Multi-step baselines are from Petersen et al. (2026). **Bold** marks the best value at each NFE (1 or 2).

| | | Molecular quality (%) | | | | Distributional similarity | | |
|---|---|---|---|---|---|---|---|---|
| **Method** | **NFE** | Valid ↑ | V.U. ↑ | V.U.N. ↑ | Filters ↑ | FCD ↓ | SNN ↑ | Scaf. ↑ |
| Training set | – | 100.0 | 100.0 | 0.0 | 100.0 | 0.01 | 0.64 | 99.1 |
| *Multi-step references* | | | | | | | | |
| DiGress | 500 | 85.7 | 85.7 | 81.4 | 97.1 | 1.19 | 0.52 | 14.8 |
| DisCo | 500 | 88.3 | 88.3 | 86.3 | 95.6 | 1.44 | 0.50 | 15.1 |
| Cometh | 500 | 90.5 | 90.4 | 83.7 | 99.1 | 1.27 | 0.54 | 16.0 |
| DeFoG | 50 | 83.9 | 83.8 | 81.2 | 96.5 | 1.87 | 0.50 | 23.5 |
| DeFoG | 500 | 92.8 | 92.7 | 85.4 | 98.9 | 1.95 | 0.55 | 14.4 |
| GraphBSI | 10 | 32.8 | 31.1 | 30.7 | 86.4 | 7.93 | 0.45 | 22.3 |
| GraphBSI | 50 | 45.2 | 44.4 | 43.7 | 88.1 | 5.14 | 0.47 | 22.2 |
| *One-step generation* (NFE = 1) | | | | | | | | |
| CFM | 1 | 2.6 | 2.6 | 2.6 | 51.4 | 18.43 | **0.34** | 0.9 |
| BFM (ours) | 1 | **30.6** | **22.7** | **22.7** | **76.0** | **14.10** | 0.32 | **9.2** |
| *Two-step generation* (NFE = 2) | | | | | | | | |
| CFM | 2 | 27.9 | 27.9 | 27.9 | 45.9 | 10.65 | 0.36 | 2.4 |
| BFM (ours) | 2 | **29.6** | **28.4** | **28.1** | **75.6** | **8.40** | **0.42** | **16.4** |

Table 3: Tree graph generation. Multi-step baselines are from Qin et al. (2025). **Bold** marks the best value at each NFE (1 or 2).

| | | Graph quality (%) | | | | Distributional similarity | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **NFE** | Valid ↑ | Unique ↑ | Novel ↑ | V.U.N. ↑ | Deg. ↓ | Clus. ↓ | Orbit ↓ | Spec. ↓ | Wavelet ↓ | Ratio ↓ |
| Training set | – | 100.0 | 100.0 | 0.0 | 0.0 | 0.0001 | 0.0000 | 0.0000 | 0.0075 | 0.0030 | 1.0 |
| *Multi-step references* | | | | | | | | | | | |
| HSpectre | 256 | 100.0 | 100.0 | 100.0 | 100.0 | 0.0001 | 0.0000 | 0.0000 | 0.0117 | 0.0047 | 4.0 |
| DiGress | 1000 | 90.0 | 100.0 | 100.0 | 90.0 | 0.0002 | 0.0000 | 0.0000 | 0.0113 | 0.0043 | 1.6 |
| DeFoG | 50 | 68.0 | 100.0 | 100.0 | 68.0 | 0.0002 | 0.0000 | 0.0000 | 0.0115 | 0.0046 | 1.7 |
| DeFoG | 1000 | 96.5 | 100.0 | 100.0 | 96.5 | 0.0002 | 0.0000 | 0.0000 | 0.0108 | 0.0046 | 1.6 |
| GraphBSI | 10 | 6.5 | 100.0 | 100.0 | 6.5 | 0.0007 | 0.0020 | 0.0006 | 0.0142 | 0.0086 | 3.8 |
| GraphBSI | 50 | 55.0 | 100.0 | 100.0 | 55.0 | 0.0003 | 0.0025 | 0.0001 | 0.0111 | 0.0047 | 1.9 |
| *One-step generation* (NFE = 1) | | | | | | | | | | | |
| CFM | 1 | 1.0 | **100.0** | **100.0** | 1.0 | **0.0004** | 0.0128 | **0.0001** | 0.0167 | 0.0098 | 3.2 |
| BFM (ours) | 1 | **6.0** | **100.0** | **100.0** | **6.0** | **0.0004** | **0.0036** | 0.0006 | **0.0120** | **0.0068** | **2.5** |
| *Two-step generation* (NFE = 2) | | | | | | | | | | | |
| CFM | 2 | 2.5 | **100.0** | **100.0** | 2.5 | 0.0005 | 0.0136 | **0.0002** | **0.0113** | **0.0059** | 2.9 |
| BFM (ours) | 2 | **11.5** | **100.0** | **100.0** | **11.5** | **0.0004** | **0.0037** | 0.0004 | 0.0124 | 0.0061 | **2.6** |

**MOSES**. BFM's one-step advantage over CFM is largest on MOSES. In Table 2, CFM's one-step validity nearly collapses while BFM retains high validity. BFM's second step further improves V.U.N., FCD, and Scaf., with a slight decrease in validity and filter pass rate.

**Tree**. Table 3 shows that BFM produces several times the V.U.N. of CFM at both one and two evaluations and also attains a lower aggregate MMD ratio at both budgets. Although CFM attains lower spectral and wavelet MMDs at two steps, it produces about a fifth as many valid trees.

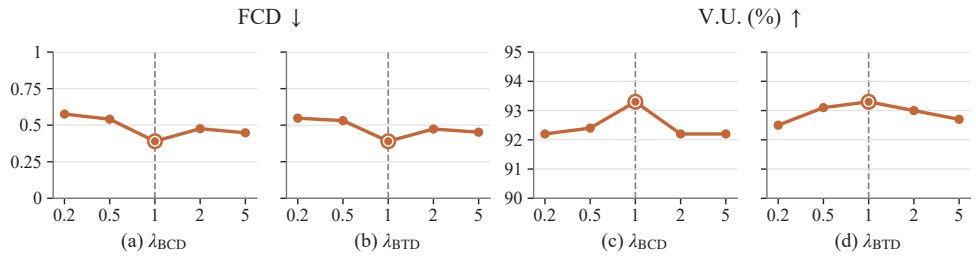


Figure 3: **One-step loss-weight sensitivity on QM9. (a,b)** FCD; **(c,d)** V.U. Each sweep fixes the other weight at 1; dashed lines mark the default. $\lambda = 1$ is optimal in both sweeps.

### 3.3 Ablation Study

Table 4 isolates each objective on QM9. BCD alone achieves much lower one-step FCD than BTD alone, consistent with its strength on long steps and confirming that tangent supervision alone cannot accurately predict long belief steps. However, the two-step FCD of BCD alone does not improve over one step, indicating that chord supervision alone does not compose well across shorter intervals. Adding BTD addresses this: the full model achieves the best FCD at both budgets and maintains the highest V.U.

Figure 3 sweeps $\lambda_{\text{BCD}}$ and $\lambda_{\text{BTD}}$ for one-step sampling. Generation quality remains relatively stable, suggesting that BFM is robust to the choice of loss weights. Multi-step results are in Appendix B.1.

Table 4: Ablation of BCD and BTD on QM9.

| | 1 step | | 2 steps | |
|---|---|---|---|---|
| **Method** | FCD ↓ | V.U. ↑ | FCD ↓ | V.U. ↑ |
| BCD only | 0.57 | 93.2 | 0.61 | 92.9 |
| BTD only | 1.90 | 91.2 | 1.00 | **94.0** |
| BCD + BTD | **0.39** | **93.3** | **0.34** | **94.0** |

## 4 Related Work

**Bayesian flow networks and sample inference**. Bayesian Flow Networks (BFN) generate data via Bayesian parameter updates (Graves et al., 2023) and have been extended to 3D molecules, crystals, and discrete graphs (Song et al., 2024; Wu et al., 2025; Song et al., 2025), the last interpolating topological information through continuous counts. Bayesian Sample Inference (BSI) formulates generation as posterior inference, with GraphBSI adapting it to categorical graphs (Lienen et al., 2026; Petersen et al., 2026). Within the BFN family, Xue et al. (2024) derive efficient samplers through a connection to diffusion. BFM builds on BSI for one-step and two-step graph generation.

**Consistency models and flow maps**. Consistency models and flow map self-distillation replace multistep numerical integration with learned transformations for continuous data (Song et al., 2023; Boffi et al., 2025). Categorical Flow Maps extend finite-time map learning to categorical data (Roos et al., 2026), and BFM further extends it to BSI's Bayesian inference setting.

## 5 Conclusion

We introduced Belief Flow Maps (BFM), a distillation framework that compresses the multi-step sampling of Bayesian Sample Inference into one or two network evaluations. The key idea is that, in normalized coordinates, the probability flow of BSI transports each belief along a data-dependent trajectory, so several local steps along this flow can serve as the teacher for a single transition. Belief Chord Distillation provides global supervision: it distills the belief reached by these steps, so the whole transition from the source belief follows the trajectory. Belief Tangent Distillation provides complementary local supervision: it aligns the derivative of the learned transition at the belief it reaches with the belief flow velocity. Together, the two objectives constrain both where a transition ends and how it arrives there. On four datasets, BFM achieves the best distributional fidelity among compared methods with one or two network evaluations.

### Reproducibility Statement

All implementation details, hyperparameters, and training procedures are described in Appendix A.8. The precision schedule, predictor parameterization, and distillation targets are derived in Appendices A.1–A.2. We report dataset splits, evaluation protocols, sample sizes, and metrics for each benchmark in §3.1 and Appendix A.7. Code will be made available upon acceptance.

### AI Use Disclosure

In this work, we used generative AI tools (large language models) to assist with editing the paper for readability. We did not use generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for or write proofs, propose or refine hypotheses, design methodology or experiments, implement methods, generate datasets, interpret results, or create scientific figures. All such tasks were performed entirely by the authors. We have reviewed all AI-assisted text for correctness and take responsibility for the final content of this work, including all text, claims, and artifacts.

## References


Guy W. Bemis and Mark A. Murcko. The properties of known drugs. 1. Molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *ICLR*, 2024.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. In *NeurIPS*, volume 38, 2025.

Yuanqi Du, Arian R. Jamasb, Jeff Guo, Tianfan Fu, Charles Harris, Yingheng Wang, Chenru Duan, Pietro Liò, Philippe Schwaller, and Tom L. Blundell. Machine learning-aided generative molecular design. *Nature Machine Intelligence*, 6(6):589–604, 2024.

Floor Eijkelboom, Grigory Bartosh, Christian A. Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *NeurIPS*, volume 37, pp. 11735–11764, 2024.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Alex Graves, Rupesh Kumar Srivastava, Timothy Atkinson, and Faustino Gomez. Bayesian flow networks. *arXiv preprint arXiv:2308.07037*, 2023.

Jeff Guo and Philippe Schwaller. TANGO: direct optimization of constrained synthesizability for generative molecular design. *Nature Computational Science*, 6(3):260–270, 2026.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *ICML*, volume 162, pp. 10362–10383, 2022.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. In *ICML*, volume 235, pp. 22371–22405, 2024.

Greg Landrum. RDKit: Open-source cheminformatics, 2006. URL `https://www.rdkit.org`.

Marten Lienen, Marcel Kollovieh, and Stephan Günnemann. Generative modeling with Bayesian sample inference. *Transactions on Machine Learning Research*, 2026.

Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *NeurIPS*, volume 31, pp. 7113–7124, 2018.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. SPECTRE: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *ICML*, volume 162, pp. 15159–15179, 2022.

Ole Petersen, Marcel Kollovieh, Marten Lienen, and Stephan Günnemann. Discrete Bayesian sample inference for graph generation. In *ICLR*, 2026.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular Sets (MOSES): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11:565644, 2020.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet ChemNet distance: A metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 58(9):1736–1741, 2018.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *ICML*, volume 267, pp. 50269–50326, 2025.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael M. Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps. In *ICML*, 2026.

Arne Schneuing, Charles Harris, Yuanqi Du, Kieran Didi, Arian Jamasb, Ilia Igashov, Weitao Du, Carla Gomes, Tom L. Blundell, Pietro Lio, Max Welling, Michael Bronstein, and Bruno Correia. Structure-based drug design with equivariant diffusion models. *Nature Computational Science*, 4 (12):899–909, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, volume 202, pp. 32211–32252, 2023.

Yuxuan Song, Jingjing Gong, Hao Zhou, Mingyue Zheng, Jingjing Liu, and Wei-Ying Ma. Unified generative modeling of 3D molecules with Bayesian flow networks. In *ICLR*, 2024.

Yuxuan Song, Juntong Shi, Jingjing Gong, Minkai Xu, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Smooth interpolation for improved discrete graph generative models. In *ICML*, volume 267, pp. 56363–56388, 2025.

Teague Sterling and John J. Irwin. ZINC 15–ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *ICLR*, 2023.

Hanlin Wu, Yuxuan Song, Jingjing Gong, Ziyao Cao, Yawen Ouyang, Jianbing Zhang, Hao Zhou, Wei-Ying Ma, and Jingjing Liu. A periodic Bayesian flow for material generation. In *ICLR*, 2025.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

Kaiwen Xue, Yuhao Zhou, Shen Nie, Xu Min, Xiaolu Zhang, Jun Zhou, and Chongxuan Li. Unifying Bayesian flow networks and diffusion models through stochastic differential equations. In *ICML*, volume 235, pp. 55656–55681, 2024.

Chengxi Zang and Fei Wang. MoFlow: An invertible flow model for generating molecular graphs. In *KDD*, pp. 617–626, 2020.

# A Additional Method Details

## A.1 SDEs, the Fokker–Planck Equation, and Probability Flow ODEs

This section gives the steps behind the passage from the belief SDE to its probability flow ODE in §2.2. We first recall the general relation between an SDE, its Fokker–Planck equation, and its probability flow ODE (Song et al., 2021), and then specialize it to the belief dynamics.

**Notation.** Throughout, $\beta_t = \beta(t)$ is the cumulative precision schedule of Eq. 69, which satisfies $\beta_0 = 0$, and $\beta'_t = \mathrm{d}\beta_t/\mathrm{d}t$ is its rate. For this schedule, with $r = \log(\beta_{\mathrm{end}}/\beta_{\mathrm{start}}) > 0$,

$$\beta_t = \beta_{\mathrm{start}}\big(e^{rt} - 1\big), \qquad \beta'_t = \beta_{\mathrm{start}}\, r\, e^{rt} > 0. \tag{24}$$

The prior is $\boldsymbol{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$ with a constant $\sigma_0^2 > 0$. As in Eq. 12, we write

$$\sigma_t^2 = \sigma_0^2 + \beta_t, \qquad \tau_t = \frac{\beta_t}{\sigma_t}, \qquad \widetilde{\boldsymbol{z}}_t = \frac{\boldsymbol{z}_t - \boldsymbol{\mu}_0}{\sigma_t}. \tag{25}$$

Since $\beta_0 = 0$, $\sigma_0$ is the prior standard deviation and $\tau_0 = 0$.

**Itô SDE.** Consider an Itô process on $\mathbb{R}^D$ defined by the stochastic differential equation

$$\mathrm{d}\boldsymbol{z}_t = \boldsymbol{f}_t\, \mathrm{d}t + g_t\, \mathrm{d}\boldsymbol{w}_t, \tag{26}$$

where $\boldsymbol{z}_t$ is the state at time $t$, $\boldsymbol{f}_t = \boldsymbol{f}(\boldsymbol{z}_t, t)$ is the drift coefficient, $g_t = g(t)$ is a state-independent diffusion coefficient, and $\boldsymbol{w}_t$ is a standard Wiener process. Let $q_t = q_t(\boldsymbol{z}_t)$ denote the density of $\boldsymbol{z}_t$.

**Fokker–Planck equation.** For an Itô SDE with a state-independent diffusion coefficient, the density of Eq. 26 evolves according to the Fokker–Planck equation

$$\frac{\partial q_t}{\partial t} = -\nabla_{\boldsymbol{z}_t} \cdot \big(\boldsymbol{f}_t\, q_t\big) + \frac{g_t^2}{2}\, \Delta q_t, \tag{27}$$

where $\Delta = \nabla_{\boldsymbol{z}_t} \cdot \nabla_{\boldsymbol{z}_t}$ is the Laplacian. The diffusion term can be written as a divergence. Since $\nabla_{\boldsymbol{z}_t} \log q_t = \nabla_{\boldsymbol{z}_t} q_t / q_t$, we have $\nabla_{\boldsymbol{z}_t} q_t = q_t \nabla_{\boldsymbol{z}_t} \log q_t$ and therefore

$$\Delta q_t = \nabla_{\boldsymbol{z}_t} \cdot \big(\nabla_{\boldsymbol{z}_t} q_t\big) = \nabla_{\boldsymbol{z}_t} \cdot \big(q_t \nabla_{\boldsymbol{z}_t} \log q_t\big). \tag{28}$$

Substituting this into Eq. 27 and collecting both terms under one divergence gives

$$\frac{\partial q_t}{\partial t} = -\nabla_{\boldsymbol{z}_t} \cdot \big(\boldsymbol{f}_t\, q_t\big) + \frac{g_t^2}{2}\, \nabla_{\boldsymbol{z}_t} \cdot \big(q_t \nabla_{\boldsymbol{z}_t} \log q_t\big) = -\nabla_{\boldsymbol{z}_t} \cdot \Big[\Big(\boldsymbol{f}_t - \frac{1}{2} g_t^2 \nabla_{\boldsymbol{z}_t} \log q_t\Big) q_t\Big]. \tag{29}$$

**Probability flow ODE.** The last form of Eq. 29 is itself the Fokker–Planck equation of another process. For any process $\mathrm{d}\boldsymbol{z}_t = \widetilde{\boldsymbol{f}}_t\, \mathrm{d}t + \widetilde{g}_t\, \mathrm{d}\boldsymbol{w}_t$, Eq. 27 reads $\partial_t q_t = -\nabla_{\boldsymbol{z}_t} \cdot (\widetilde{\boldsymbol{f}}_t q_t) + \frac{\widetilde{g}_t^2}{2} \Delta q_t$. Choosing the new drift and diffusion coefficients as

$$\widetilde{\boldsymbol{f}}_t = \boldsymbol{f}_t - \frac{1}{2} g_t^2 \nabla_{\boldsymbol{z}_t} \log q_t, \qquad \widetilde{g}_t = 0, \tag{30}$$

recovers exactly the last form of Eq. 29. The resulting process has no noise, so it is the ordinary differential equation

$$\mathrm{d}\boldsymbol{z}_t = \Big(\boldsymbol{f}_t - \frac{1}{2} g_t^2 \nabla_{\boldsymbol{z}_t} \log q_t\Big) \mathrm{d}t. \tag{31}$$

Eqs. 26 and 31 share the same Fokker–Planck equation. Assuming this equation has a unique solution for a given initial density, starting Eq. 31 from $\boldsymbol{z}_0 \sim q_0$ produces the same density $q_t$ as the SDE at every time $t$, although each trajectory is deterministic. Eq. 31 is the probability flow ODE of Eq. 26.

**Belief SDE and its marginal.** For a fixed graph $\boldsymbol{x}$, the belief SDE in Eq. 6 is Eq. 26 with $\boldsymbol{f}_t = \beta'_t \boldsymbol{x}$ and $g_t = \sqrt{\beta'_t}$, so Eq. 27 gives the conditional Fokker–Planck equation in Eq. 7. During generation the graph $\boldsymbol{x} \sim p(\boldsymbol{x})$ is unknown, so the relevant density is the marginal

$$q_t(\boldsymbol{z}) = \int q_t(\boldsymbol{z}|\boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \tag{32}$$

where the integral is a sum over graphs, since graphs are discrete. Let $q_t(\boldsymbol{x}|\boldsymbol{z}) = p(\boldsymbol{x})\, q_t(\boldsymbol{z}|\boldsymbol{x}) / q_t(\boldsymbol{z})$ denote the posterior induced by the data distribution and the Gaussian encoding, which generally

differs from the per-category softmax belief $p(x_k|\boldsymbol{z})$ of §2.1, and let $\boldsymbol{x}_t^\star(\boldsymbol{z}) = \mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t = \boldsymbol{z}] = \int \boldsymbol{x}\, q_t(\boldsymbol{x}|\boldsymbol{z})\,\mathrm{d}\boldsymbol{x}$ denote its mean. Since $p(\boldsymbol{x})$ does not depend on $t$ and the diffusion term is linear in the density, averaging Eq. 7 over $p(\boldsymbol{x})$ gives

$$\frac{\partial q_t(\boldsymbol{z})}{\partial t} = -\nabla_{\boldsymbol{z}} \cdot \Big[\beta_t' \int \boldsymbol{x}\, q_t(\boldsymbol{z}|\boldsymbol{x})\, p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\Big] + \frac{\beta_t'}{2}\Delta q_t(\boldsymbol{z}). \tag{33}$$

The key step is the drift term. By Bayes' rule $q_t(\boldsymbol{z}|\boldsymbol{x})\, p(\boldsymbol{x}) = q_t(\boldsymbol{z})\, q_t(\boldsymbol{x}|\boldsymbol{z})$,

$$\int \boldsymbol{x}\, q_t(\boldsymbol{z}|\boldsymbol{x})\, p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = q_t(\boldsymbol{z}) \int \boldsymbol{x}\, q_t(\boldsymbol{x}|\boldsymbol{z})\,\mathrm{d}\boldsymbol{x} = q_t(\boldsymbol{z})\, \boldsymbol{x}_t^\star(\boldsymbol{z}). \tag{34}$$

Substituting this into Eq. 33 shows that the marginal density satisfies Eq. 27 with drift $\boldsymbol{f}_t = \beta_t' \boldsymbol{x}_t^\star(\boldsymbol{z})$ and $g_t = \sqrt{\beta_t'}$, and Eq. 31 gives the marginal probability flow ODE in Eq. 9 with $\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t] = \boldsymbol{x}_t^\star(\boldsymbol{z}_t)$.

**Conditional density and score.** Because the drift $\beta_t'\boldsymbol{x}$ does not depend on $\boldsymbol{z}_t$, the belief SDE can be integrated directly. Using $\beta_0 = 0$,

$$\boldsymbol{z}_t = \boldsymbol{z}_0 + \int_0^t \beta_u' \boldsymbol{x}\,\mathrm{d}u + \int_0^t \sqrt{\beta_u'}\,\mathrm{d}\boldsymbol{w}_u = \boldsymbol{z}_0 + \beta_t \boldsymbol{x} + \int_0^t \sqrt{\beta_u'}\,\mathrm{d}\boldsymbol{w}_u. \tag{35}$$

The stochastic integral is Gaussian with zero mean, and by the Itô isometry its covariance is $\int_0^t \beta_u'\,\mathrm{d}u\, I = \beta_t I$. The prior sample $\boldsymbol{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$ is drawn independently of the graph $\boldsymbol{x}$ and of the Wiener process, so, given $\boldsymbol{x}$, the means and covariances add:

$$q_t(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{\mu}_0 + \beta_t \boldsymbol{x},\ \sigma_t^2 I\big), \qquad \sigma_t^2 = \sigma_0^2 + \beta_t, \tag{36}$$

which agrees with the discrete Bayesian update in Eq. 4. For a Gaussian $\mathcal{N}(\boldsymbol{m}, \sigma^2 I)$, the log density is $-\|\boldsymbol{z} - \boldsymbol{m}\|^2/(2\sigma^2)$ up to a constant, so its gradient is $-(\boldsymbol{z} - \boldsymbol{m})/\sigma^2$. The conditional score is therefore

$$\nabla_{\boldsymbol{z}} \log q_t(\boldsymbol{z}|\boldsymbol{x}) = -\frac{\boldsymbol{z} - \boldsymbol{\mu}_0 - \beta_t \boldsymbol{x}}{\sigma_t^2}. \tag{37}$$

**Marginal score.** Differentiating $q_t(\boldsymbol{z})$ under the integral and using $\nabla_{\boldsymbol{z}} q_t(\boldsymbol{z}|\boldsymbol{x}) = q_t(\boldsymbol{z}|\boldsymbol{x})\nabla_{\boldsymbol{z}} \log q_t(\boldsymbol{z}|\boldsymbol{x})$, the marginal score is the posterior average of the conditional score. Since the conditional score is affine in $\boldsymbol{x}$,

$$\nabla_{\boldsymbol{z}} \log q_t(\boldsymbol{z}) = \mathbb{E}_{q_t(\boldsymbol{x}|\boldsymbol{z})}\big[\nabla_{\boldsymbol{z}} \log q_t(\boldsymbol{z}|\boldsymbol{x})\big] = -\frac{\boldsymbol{z} - \boldsymbol{\mu}_0 - \beta_t\, \boldsymbol{x}_t^\star(\boldsymbol{z})}{\sigma_t^2}. \tag{38}$$

Substituting it into Eq. 9 and collecting the terms in $\boldsymbol{x}_t^\star$ gives the exact marginal probability flow ODE

$$\frac{\mathrm{d}\boldsymbol{z}_t}{\mathrm{d}t} = \beta_t'\Big(1 - \frac{\beta_t}{2\sigma_t^2}\Big)\boldsymbol{x}_t^\star(\boldsymbol{z}_t) + \frac{\beta_t'}{2\sigma_t^2}\big(\boldsymbol{z}_t - \boldsymbol{\mu}_0\big). \tag{39}$$

**Exact posterior and learned approximation.** All identities above hold for the exact posterior mean $\boldsymbol{x}_t^\star$, which is intractable. The reconstructor $\boldsymbol{x}^\theta(\boldsymbol{z}, t)$ is trained with a squared reconstruction loss. At each time $t$ and belief $\boldsymbol{z}$, a squared loss is minimized pointwise by the conditional mean $\boldsymbol{x}_t^\star(\boldsymbol{z})$. The graph norm weights node and edge components by $1/|\mathcal{V}|$ and $\lambda_E/\max(|\mathcal{E}^+|, 1)$, which depend only on the number of nodes. Since the number of nodes and the validity masks are fixed by the input, these weights do not depend on the labels being predicted, and a positive time weighting does not change the minimizer at each $t$, so the same holds for the reconstruction loss. The reconstructor therefore serves as an approximation of the posterior mean. Replacing $\boldsymbol{x}_t^\star$ by $\boldsymbol{x}^\theta$ in Eqs. 38 and 39 gives the learned marginal score in Eq. 10 and the learned belief flow velocity in Eq. 11. Replacing the exact posterior mean by the reconstructor generally loses exact preservation of the marginals $q_t$, while the change of coordinates below remains exact for the learned field.

**Normalized coordinates.** Since $\sigma_t = (\sigma_0^2 + \beta_t)^{1/2}$ and $\sigma_0$ is constant,

$$\sigma_t' = \frac{1}{2}\big(\sigma_0^2 + \beta_t\big)^{-1/2}\beta_t' = \frac{\beta_t'}{2\sigma_t}. \tag{40}$$

Applying the quotient rule to $\tau_t = \beta_t/\sigma_t$ and substituting $\sigma_t'$,

$$\tau_t' = \frac{\beta_t'\sigma_t - \beta_t\sigma_t'}{\sigma_t^2} = \frac{\beta_t'\sigma_t - \beta_t\beta_t'/(2\sigma_t)}{\sigma_t^2} = \frac{\beta_t'(\sigma_t^2 - \beta_t/2)}{\sigma_t^3} = \frac{\beta_t'(\sigma_0^2 + \beta_t/2)}{\sigma_t^3}, \tag{41}$$

where the last step uses $\sigma_t^2 = \sigma_0^2 + \beta_t$. Since $\beta_t' > 0$ (Eq. 24) and $\sigma_0^2 + \beta_t/2 > 0$, we have $\tau_t' > 0$. Hence $\tau_t$ is strictly increasing in $t$, the map $t \mapsto \tau_t$ is invertible, and $\tau_t$ can be used as the time variable.

Next, since $\boldsymbol{\mu}_0$ is constant, the quotient rule applied to $\widetilde{\boldsymbol{z}}_t = (\boldsymbol{z}_t - \boldsymbol{\mu}_0)/\sigma_t$ gives

$$\frac{\mathrm{d}\widetilde{\boldsymbol{z}}_t}{\mathrm{d}t} = \frac{1}{\sigma_t}\frac{\mathrm{d}\boldsymbol{z}_t}{\mathrm{d}t} - \frac{\sigma_t'}{\sigma_t^2}\big(\boldsymbol{z}_t - \boldsymbol{\mu}_0\big), \qquad \frac{\sigma_t'}{\sigma_t^2} = \frac{\beta_t'}{2\sigma_t^3}. \tag{42}$$

Substituting Eq. 39,

$$\begin{aligned}\frac{\mathrm{d}\widetilde{\boldsymbol{z}}_t}{\mathrm{d}t} &= \frac{\beta_t'}{\sigma_t}\Big(1 - \frac{\beta_t}{2\sigma_t^2}\Big)\boldsymbol{x}_t^\star(\boldsymbol{z}_t) + \frac{\beta_t'}{2\sigma_t^3}\big(\boldsymbol{z}_t - \boldsymbol{\mu}_0\big) - \frac{\beta_t'}{2\sigma_t^3}\big(\boldsymbol{z}_t - \boldsymbol{\mu}_0\big) \\ &= \frac{\beta_t'(\sigma_t^2 - \beta_t/2)}{\sigma_t^3}\,\boldsymbol{x}_t^\star(\boldsymbol{z}_t) = \tau_t'\,\boldsymbol{x}_t^\star(\boldsymbol{z}_t),\end{aligned} \tag{43}$$

where the terms proportional to $\boldsymbol{z}_t - \boldsymbol{\mu}_0$ cancel and the remaining coefficient equals $\tau_t'$ in Eq. 41. Since $\tau_t' > 0$, the chain rule gives $\mathrm{d}\widetilde{\boldsymbol{z}}_t/\mathrm{d}\tau_t = (\mathrm{d}\widetilde{\boldsymbol{z}}_t/\mathrm{d}t)/(\mathrm{d}\tau_t/\mathrm{d}t)$. Writing $\boldsymbol{z}_t = \boldsymbol{\mu}_0 + \sigma_t\widetilde{\boldsymbol{z}}_t$ in the argument of $\boldsymbol{x}_t^\star$ yields the normalized belief ODE

$$\frac{\mathrm{d}\widetilde{\boldsymbol{z}}_t}{\mathrm{d}\tau_t} = \boldsymbol{x}_t^\star\big(\boldsymbol{\mu}_0 + \sigma_t\widetilde{\boldsymbol{z}}_t\big). \tag{44}$$

Replacing $\boldsymbol{x}_t^\star$ by $\boldsymbol{x}^\theta$ gives the learned normalized belief ODE in Eq. 13.

**Conditional straight lines and marginal curvature.** For a fixed graph $\boldsymbol{x}$, Eq. 31 with drift $\beta_t'\boldsymbol{x}$ and the conditional score above give the conditional probability flow ODE

$$\frac{\mathrm{d}\boldsymbol{z}_t}{\mathrm{d}t} = \beta_t'\boldsymbol{x} + \frac{\beta_t'}{2\sigma_t^2}\big(\boldsymbol{z}_t - \boldsymbol{\mu}_0 - \beta_t\boldsymbol{x}\big) = \beta_t'\Big(1 - \frac{\beta_t}{2\sigma_t^2}\Big)\boldsymbol{x} + \frac{\beta_t'}{2\sigma_t^2}\big(\boldsymbol{z}_t - \boldsymbol{\mu}_0\big), \tag{45}$$

which is Eq. 39 with $\boldsymbol{x}_t^\star(\boldsymbol{z}_t)$ replaced by the fixed graph $\boldsymbol{x}$. The same computation therefore gives $\mathrm{d}\widetilde{\boldsymbol{z}}_t/\mathrm{d}\tau_t = \boldsymbol{x}$. Since $\boldsymbol{x}$ does not depend on $\tau$, integrating from $\tau_s$ to $\tau_t$ yields

$$\widetilde{\boldsymbol{z}}_t = \widetilde{\boldsymbol{z}}_s + (\tau_t - \tau_s)\,\boldsymbol{x}, \tag{46}$$

so each conditional trajectory is exactly a straight line in normalized coordinates. For the marginal dynamics, the normalization removes the explicit term in $\boldsymbol{z}_t - \boldsymbol{\mu}_0$, but $\boldsymbol{x}_t^\star(\boldsymbol{\mu}_0 + \sigma_t\widetilde{\boldsymbol{z}}_t)$ still depends on the state, and so does its learned approximation. Eq. 44 is therefore not a linear ODE in general, and its trajectories bend as the posterior mean changes along the way. This curvature is what a single large transition must account for, and it motivates learning the interval average in Eq. 15 rather than a single belief flow velocity evaluation.

### A.2 Parameterization and Training

**Predictor parameterization.** A shared graph network $f_\theta$ outputs residual category logits per node and edge block, and a blockwise softmax maps them to interval-average probabilities:

$$\boldsymbol{x}_{s,t}^\theta(\boldsymbol{z}_s) = \mathrm{softmax}\big(\boldsymbol{z}_s + f_\theta(\boldsymbol{z}_s, s, t)\big)\,. \tag{47}$$

At equal times this reduces to the reconstructor, $\boldsymbol{x}_{t,t}^\theta(\boldsymbol{z}) = \boldsymbol{x}^\theta(\boldsymbol{z}, t)$, so predictor and reconstructor share a single network. Converting the normalized coordinate map (Eq. 14) back to the original logit coordinates gives the z-space update in Eq. 21. We derive this by substituting $\widetilde{\boldsymbol{z}}_t = (\boldsymbol{z}_t - \boldsymbol{\mu}_0)/\sigma_t$ (Eq. 12) into both sides of Eq. 14:

$$\frac{\boldsymbol{z}_t^\theta - \boldsymbol{\mu}_0}{\sigma_t} = \frac{\boldsymbol{z}_s - \boldsymbol{\mu}_0}{\sigma_s} + (\tau_t - \tau_s)\,\boldsymbol{x}_{s,t}^\theta. \tag{48}$$

Multiplying both sides by $\sigma_t$ and writing $a_{s,t} = \sigma_t/\sigma_s$,

$$\boldsymbol{z}_t^\theta - \boldsymbol{\mu}_0 = a_{s,t}(\boldsymbol{z}_s - \boldsymbol{\mu}_0) + \sigma_t(\tau_t - \tau_s)\,\boldsymbol{x}_{s,t}^\theta. \tag{49}$$

From $\tau_t = \beta_t/\sigma_t$, the coefficient $\sigma_t(\tau_t - \tau_s) = \beta_t - a_{s,t}\beta_s$. Adding $\boldsymbol{\mu}_0$ to both sides recovers Eq. 21.

**Training distribution.** Beliefs conditioned on a training graph follow the Gaussian $q_t(\boldsymbol{z}|\boldsymbol{x})$ of Appendix A.1, so training beliefs can be sampled as

$$\boldsymbol{z}_t = \boldsymbol{\mu}_0 + \beta_t \boldsymbol{x} + \sigma_t \, \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I). \tag{50}$$

Noise is independent across category blocks for valid nodes and unique undirected edges, and edge values are mirrored in the symmetric tensor.

**Graph norm.** For a graph with valid nodes $\mathcal{V}$ and candidate edges $\mathcal{E}^+ = \{(i,j) : i < j,\, i,j \in \mathcal{V}\}$, the squared graph norm of a residual $\mathbf{r} = (\mathbf{r}^X, \mathbf{r}^E)$ averages node and edge errors separately:

$$\|\mathbf{r}\|_G^2 = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \|\mathbf{r}_i^X\|_2^2 + \lambda_E \frac{1}{\max(|\mathcal{E}^+|, 1)} \sum_{(i,j) \in \mathcal{E}^+} \|\mathbf{r}_{ij}^E\|_2^2. \tag{51}$$

The weight $\lambda_E$ balances edge errors against node errors, and separate averaging prevents graph size from changing either contribution.

**BCD integration.** The BCD target in Algorithm 3 uses $d = 4$ Heun substeps as listed in Table 5, and all targets are detached. The grid is uniformly spaced in signal-to-noise time, and the grid times $t_j$ are obtained through the inverse of $t \mapsto \tau_t$.

**BTD derivatives.** BTD (§2.4) differentiates the normalized map with respect to signal-to-noise time $\tau_t$, using the chain rule $\partial_{\tau_t} = {\tau_t'}^{-1} \partial_t$ with $\tau_t'$ from Eq. 41. Holding the source state and time fixed, the derivative of the normalized map $\widetilde{\boldsymbol{z}}_{s,t}^\theta = \widetilde{\boldsymbol{z}}_s + (\tau_t - \tau_s)\boldsymbol{x}_{s,t}^\theta$ (Eq. 14) with respect to $\tau_t$ is

$$\partial_{\tau_t} \widetilde{\boldsymbol{z}}_{s,t}^\theta = \boldsymbol{x}_{s,t}^\theta + (\tau_t - \tau_s) \partial_{\tau_t} \boldsymbol{x}_{s,t}^\theta. \tag{52}$$

**Time-pair sampling.** The distribution over $(s,t)$ includes endpoint pairs with $t = 1$, interior intervals, and full-horizon pairs $(s,t) = (0,1)$. Full-horizon pairs train the one-step map, and shorter intervals train the maps composed during few-step sampling.

### A.3 Training and Sampling Algorithms

Algorithm 1 summarizes BFM training, Algorithm 2 summarizes few-step sampling, and Algorithm 3 computes the BCD target used in training.

### A.4 Terminal-Head Distillation

**Motivation.** In BSI, the reconstructor $\boldsymbol{x}^\theta(\boldsymbol{z}_t, t)$ approximates the posterior mean $\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t]$ under a squared-error loss (Lienen et al., 2026; Petersen et al., 2026). The GNN backbone processes all node and edge beliefs jointly, so its intermediate hidden features $\boldsymbol{H}$ capture graph-wide structural correlations, such as ring membership, valence constraints, and connectivity patterns, that extend beyond the per-component logits. With few transport steps, the final belief $\boldsymbol{z}_{t_K}$ may not fully collapse onto a single category per block, making direct argmax unreliable. However, the hidden features from the last backbone evaluation already encode rich structural information about the predicted graph. The terminal head exploits this by reading $\boldsymbol{H}$ to refine the raw logits into sharper category distributions, effectively performing a lightweight denoising step without an additional backbone evaluation.

**Head parameterization.** The terminal decoder $g_\psi$ adds a learned residual to the final belief logits before a blockwise softmax:

$$g_\psi(\boldsymbol{z}_{t_K}, \boldsymbol{H}) = \mathrm{softmax}(\boldsymbol{z}_{t_K} + \widetilde{g}_\psi([\boldsymbol{H}, \boldsymbol{z}_{t_K}])) \,, \tag{53}$$

where $\boldsymbol{H}$ denotes the hidden features of the last backbone evaluation, brackets denote concatenation, and $\widetilde{g}_\psi$ is a small network that reads both.

**Head loss.** The head is distilled from the full reconstructor evaluated at the transported endpoint. Starting from a prior sample $\boldsymbol{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$, the transport map produces the endpoint logits

$$\boldsymbol{z}_1^\theta = \boldsymbol{z}_{0,1}^\theta(\boldsymbol{z}_0), \tag{54}$$

from which the full reconstructor gives the teacher distribution $\boldsymbol{x}^\theta(\boldsymbol{z}_1^\theta, 1)$. For one-hot labels the conditional mean is the vector of category probabilities, so $\boldsymbol{x}^\theta(\boldsymbol{z}_1^\theta, 1)$ is directly a valid distribution

**Algorithm 1** BFM training

**Require:** data $p(\boldsymbol{x})$, network $\boldsymbol{x}^\theta$,
schedule $\beta_t$, prior $\mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$,
learning rate $\eta$,
weights $\lambda_{\mathrm{BCD}}, \lambda_{\mathrm{BTD}}, \lambda_{\mathrm{val}}$
**Notation:** $\sigma_u = \sqrt{\sigma_0^2 + \beta_u}$, $\tau_u = \beta_u/\sigma_u$,
$\widetilde{\boldsymbol{z}}_u = (\boldsymbol{z}_u - \boldsymbol{\mu}_0)/\sigma_u$,
$\|\cdot\|_G$ as in Eq. 51
**while** not converged **do**
$\boldsymbol{x} \sim p(\boldsymbol{x})$, times $s < t$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
$\boldsymbol{z}_u \leftarrow \boldsymbol{\mu}_0 + \beta_u \boldsymbol{x} + \sigma_u \boldsymbol{\epsilon},\ u \in \{s, t\}$
$\mathcal{L}_{\mathrm{BF}} \leftarrow \beta'_u \|\boldsymbol{x}^\theta(\boldsymbol{z}_u, u) - \boldsymbol{x}\|^2/2$
$\widetilde{\boldsymbol{z}}^\theta_{s,t} \leftarrow \widetilde{\boldsymbol{z}}_s + (\tau_t - \tau_s)\, \boldsymbol{x}^\theta_{s,t}(\boldsymbol{z}_s)$
$\boldsymbol{z}^\theta_{s,t} \leftarrow \boldsymbol{\mu}_0 + \sigma_t \widetilde{\boldsymbol{z}}^\theta_{s,t}$
$\mathcal{L}_{\mathrm{BCD}} \leftarrow \|\boldsymbol{x}^\theta_{s,t}(\boldsymbol{z}_s) - \textsc{Chord}(\boldsymbol{z}_s, s, t)\|^2_G$
$\mathcal{L}_{\mathrm{BTD}} \leftarrow \|\partial_{\tau_t} \widetilde{\boldsymbol{z}}^\theta_{s,t} - \mathrm{sg}[\boldsymbol{x}^\theta(\boldsymbol{z}^\theta_{s,t}, t)]\|^2_G$
Compute $\mathcal{L}_{\mathrm{head}}$ (Eq. 55) and $\mathcal{L}_{\mathrm{val}}$ (Eq. 60)
$\mathcal{L}_{\mathrm{base}} \leftarrow \mathcal{L}_{\mathrm{BF}} + \mathcal{L}_{\mathrm{head}} + \lambda_{\mathrm{val}}\mathcal{L}_{\mathrm{val}}$
$\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{base}} + \lambda_{\mathrm{BCD}}\mathcal{L}_{\mathrm{BCD}} + \lambda_{\mathrm{BTD}}\mathcal{L}_{\mathrm{BTD}}$
$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
**end while**

**Algorithm 2** BFM sampling

**Require:** trained $\boldsymbol{x}^\theta$, steps $K$,
grid $0 = t_0 < \cdots < t_K = 1$
$n \sim p(n),\ \boldsymbol{z}_{t_0} \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$ ▷ $n$ nodes
**for** $k = 0, \ldots, K-1$ **do**
$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}^\theta_{t_k, t_{k+1}}(\boldsymbol{z}_{t_k}),\ a \leftarrow \sigma_{t_{k+1}}/\sigma_{t_k}$
$\boldsymbol{z}_{t_{k+1}} \leftarrow \boldsymbol{\mu}_0 + a(\boldsymbol{z}_{t_k} - \boldsymbol{\mu}_0)$
$+ (\beta_{t_{k+1}} - a\beta_{t_k})\, \hat{\boldsymbol{x}}$
**end for**
$G \leftarrow \mathrm{decode}(\boldsymbol{z}_{t_K})$
**return** $G$

**Algorithm 3** BCD target $\textsc{Chord}(\boldsymbol{z}_s, s, t)$

**Require:** times $s = t_0 < \cdots < t_d = t$,
$v_j(\widetilde{\boldsymbol{z}}) = \boldsymbol{x}^\theta(\boldsymbol{\mu}_0 + \sigma_{t_j}\widetilde{\boldsymbol{z}},\, t_j)$
$\widetilde{\boldsymbol{z}} \leftarrow \widetilde{\boldsymbol{z}}_s$
**for** $j = 0, \ldots, d-1,\ \delta_j = \tau_{t_{j+1}} - \tau_{t_j}$ **do**
$\bar{v}_j \leftarrow \frac{1}{2}\big[v_j(\widetilde{\boldsymbol{z}}) + v_{j+1}\big(\widetilde{\boldsymbol{z}} + \delta_j v_j(\widetilde{\boldsymbol{z}})\big)\big]$
$\widetilde{\boldsymbol{z}} \leftarrow \widetilde{\boldsymbol{z}} + \delta_j\, \bar{v}_j$
**end for**
**return** $\mathrm{sg}\big[\frac{1}{\tau_t - \tau_s} \sum_{j=0}^{d-1} \delta_j \bar{v}_j\big]$

over categories for each node and edge. The lightweight head produces the student distribution $g_\psi(\boldsymbol{z}_1^\theta, \boldsymbol{H})$ from the same endpoint logits and the last-layer hidden features (Eq. 53).

The endpoint $\boldsymbol{z}_1^\theta$ is detached so that gradients from the head loss do not backpropagate through the transport map. The distillation loss minimizes the KL divergence from the teacher to the student, averaged over the graph:

$$\mathcal{L}_{\mathrm{head}} = \mathbb{E}_{\boldsymbol{z}_0}\left[\mathrm{KL}_G\big(\mathrm{sg}[\boldsymbol{x}^\theta(\boldsymbol{z}_1^\theta, 1)] \,\|\, g_\psi(\boldsymbol{z}_1^\theta, \boldsymbol{H})\big)\right], \tag{55}$$

where sg denotes stop-gradient. The graph-averaged KL divergence $\mathrm{KL}_G$ sums per-block divergences over nodes and edges, using the same edge weight $\lambda_E$ as the reconstruction loss:

$$\mathrm{KL}_G(\boldsymbol{p}\|\widehat{\boldsymbol{p}}) = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \mathrm{KL}(\boldsymbol{p}_i^X \|\widehat{\boldsymbol{p}}_i^X) + \frac{\lambda_E}{\max(|\mathcal{E}^+|, 1)} \sum_{(i,j)\in\mathcal{E}^+} \mathrm{KL}(\boldsymbol{p}_{ij}^E \|\widehat{\boldsymbol{p}}_{ij}^E). \tag{56}$$

Here $\boldsymbol{p}_i^X$ and $\widehat{\boldsymbol{p}}_i^X$ denote the node-$i$ blocks of the two arguments, and likewise for edges. This loss trains the head to approximate the full reconstructor's output at $t = 1$, so the terminal decoding step (Eq. 23) does not require a second backbone evaluation.

### A.5 Molecular Regularization and Full Training Objective

**Valence penalty.** For molecular graphs, we add a soft penalty for local chemical consistency, following validity-aware decoder regularization (Ma et al., 2018). The goal is to penalize predicted edge distributions whose expected bond orders exceed the allowed valence of the predicted atom type.

*Step 1: sharpen edge probabilities.* Soft edge probabilities $\boldsymbol{p}_{ij}^E$ spread mass across bond types, making the expected bond order a blurred estimate of the discrete decision. We sharpen them with temperature $\tau_v < 1$ so that the expected order better reflects the most likely bond:

$$\widetilde{\boldsymbol{p}}_{ij,k}^E = \frac{(\boldsymbol{p}_{ij,k}^E)^{1/\tau_v}}{\sum_\ell (\boldsymbol{p}_{ij,\ell}^E)^{1/\tau_v}}. \tag{57}$$

*Step 2: compute expected bond order per node.* Let $o_k$ denote the bond order of edge category $k$ ($o_k = 0$ for no edge). The expected total bond order incident on node $i$ sums over all neighbors and

bond types:

$$v_i = \sum_{j \neq i} \sum_k o_k \widetilde{\boldsymbol{p}}^E_{ij,k}. \tag{58}$$

*Step 3: penalize over-valence.* Each atom type $a$ has a maximum allowed valence $c(a)$ (e.g. $c(\mathrm{C}) = 4$, $c(\mathrm{N}) = 3$). We predict the atom type from the node probabilities via $\widehat{a}_i = \arg\max_k \mathrm{sg}(\boldsymbol{p}^X_{i,k})$, using stop-gradient so the penalty only affects edge predictions. The allowed capacity is $c_i = c(\widehat{a}_i)$, and the normalized over-valence penalty is

$$\mathcal{R}_{\mathrm{val}}(\boldsymbol{p}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \left[ \frac{\max(0, v_i - c_i)}{\max(c_i, 1)} \right]^2. \tag{59}$$

The numerator $\max(0, v_i - c_i)$ is zero when the valence constraint is satisfied, and positive when the expected bond order exceeds the capacity. Dividing by $\max(c_i, 1)$ normalizes the penalty across atom types with different capacities, and squaring makes the loss smooth at zero.

**Full training objective.** We apply the valence penalty to both the reconstructor at the transported endpoint and the terminal-head predictions. For the reconstructor, the input probabilities are $\boldsymbol{p} = \boldsymbol{x}^\theta(\boldsymbol{z}_1^\theta, 1)$. For the head, the input is $\boldsymbol{p} = g_\psi(\boldsymbol{z}_1^\theta, \boldsymbol{H})$. The two valence losses are

$$\begin{aligned} \mathcal{L}^{\mathrm{BF}}_{\mathrm{val}} &= \mathbb{E}\big[\mathcal{R}_{\mathrm{val}}(\boldsymbol{x}^\theta(\boldsymbol{z}_1^\theta, 1))\big], \\ \mathcal{L}^{\mathrm{head}}_{\mathrm{val}} &= \mathbb{E}\big[\mathcal{R}_{\mathrm{val}}(g_\psi(\boldsymbol{z}_1^\theta, \boldsymbol{H}))\big]. \end{aligned} \tag{60}$$

Combining these with the main-text losses and Eq. 20, the detailed training objective is

$$\begin{aligned} \mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{BF}} + \mathcal{L}_{\mathrm{head}} + \lambda_{\mathrm{val}} \left( \mathcal{L}^{\mathrm{BF}}_{\mathrm{val}} + \mathcal{L}^{\mathrm{head}}_{\mathrm{val}} \right) \\ + \lambda_{\mathrm{BCD}} \mathcal{L}_{\mathrm{BCD}} + \lambda_{\mathrm{BTD}} \mathcal{L}_{\mathrm{BTD}}. \end{aligned} \tag{61}$$

This expands the simplified $\mathcal{L}_{\mathrm{base}}$ and $\mathcal{L}_{\mathrm{total}}$ from the main text by splitting the valence penalty into its belief flow and head components. At test time, the sampler applies Eq. 22 followed by Eq. 23.

### A.6 Evaluation Metrics

Let $\mathcal{S} = \{G_1, \ldots, G_M\}$ denote a set of $M$ generated graphs and $\mathcal{D}$ the held-out test set.

**Validity.** A molecular graph is valid if it can be parsed into a chemically consistent molecule by RDKit (Landrum, 2006). A tree graph is valid if it is connected and acyclic.

$$\mathrm{Validity} = \frac{|\{G \in \mathcal{S} : G \text{ is valid}\}|}{|\mathcal{S}|}. \tag{62}$$

**Uniqueness and Novelty.** Among valid graphs, uniqueness measures the fraction of distinct canonical forms and novelty measures the fraction absent from the training set $\mathcal{D}_{\mathrm{train}}$:

$$\mathrm{Unique} = \frac{|\{\mathrm{canon}(G) : G \in \mathcal{S}_{\mathrm{valid}}\}|}{|\mathcal{S}_{\mathrm{valid}}|}, \qquad \mathrm{Novel} = \frac{|\{G \in \mathcal{S}_{\mathrm{valid}} : \mathrm{canon}(G) \notin \mathcal{D}_{\mathrm{train}}\}|}{|\mathcal{S}_{\mathrm{valid}}|}. \tag{63}$$

**Compound rates.** V.U. and V.U.N. are the products of the corresponding rates:

$$\mathrm{V.U.} = \mathrm{Validity} \times \mathrm{Unique}, \qquad \mathrm{V.U.N.} = \mathrm{V.U.} \times \mathrm{Novel}. \tag{64}$$

**Fréchet ChemNet Distance (FCD).** FCD measures distributional similarity in the penultimate-layer activation space of ChemNet (Preuer et al., 2018). Let $(\boldsymbol{\mu}_g, \Sigma_g)$ and $(\boldsymbol{\mu}_r, \Sigma_r)$ be the mean and covariance of the activations for generated and reference molecules:

$$\mathrm{FCD} = \|\boldsymbol{\mu}_g - \boldsymbol{\mu}_r\|_2^2 + \mathrm{tr}\Big(\Sigma_g + \Sigma_r - 2(\Sigma_g \Sigma_r)^{1/2}\Big). \tag{65}$$

**Filters.** The MOSES filter pass rate is the fraction of valid generated molecules that pass all medicinal-chemistry filters defined in the MOSES benchmark (Polykovskiy et al., 2020), including molecular weight, logP, and substructure alerts.

**SNN (nearest-neighbor similarity).** For each generated molecule $G$, the Tanimoto similarity on Morgan fingerprints to its nearest neighbor in the test set is computed. SNN is the average over all valid generated molecules:

$$\mathrm{SNN} = \frac{1}{|\mathcal{S}_{\mathrm{valid}}|} \sum_{G \in \mathcal{S}_{\mathrm{valid}}} \max_{G' \in \mathcal{D}} \mathrm{Tanimoto}(\mathrm{FP}(G), \mathrm{FP}(G')). \tag{66}$$

**Scaffold similarity (Scaf.).** Scaf. is the cosine similarity between the Bemis–Murcko scaffold frequency vectors of the generated and test sets (Bemis & Murcko, 1996; Polykovskiy et al., 2020).

**Maximum mean discrepancy (MMD).** For structural graph evaluation, we compute MMD between generated and test graph descriptor distributions. Let $\phi$ extract a descriptor histogram (degree, clustering coefficient, orbit count, spectral, or wavelet features) from a graph. The squared MMD with kernel $k$ is

$$\begin{aligned}\mathrm{MMD}^2(\mathcal{S}, \mathcal{D}) = \mathbb{E}_{G,G' \in \mathcal{S}}[k(\phi(G), \phi(G'))] - 2\,\mathbb{E}_{G \in \mathcal{S}, G' \in \mathcal{D}}[k(\phi(G), \phi(G'))] \\ + \mathbb{E}_{G,G' \in \mathcal{D}}[k(\phi(G), \phi(G'))].\end{aligned} \tag{67}$$

**Ratio.** The average MMD ratio normalizes each descriptor's squared MMD against the train–test baseline (Martinkus et al., 2022):

$$\mathrm{Ratio} = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \frac{\mathrm{MMD}_f^2(\mathcal{S}, \mathcal{D})}{\mathrm{MMD}_f^2(\mathcal{D}_{\mathrm{train}}, \mathcal{D})}, \tag{68}$$

where $\mathcal{F}$ is the set of descriptors (degree, clustering, orbit, spectral, wavelet).

### A.7 Benchmark Details

All molecular datasets are encoded as typed graphs: each node carries a categorical atom type and each candidate edge carries a categorical bond type (including a no bond category). QM9 and ZINC250k use 4 edge categories (single, double, triple, no bond), while MOSES additionally includes an aromatic bond category (5 edge categories).

**QM9.** QM9 (Ramakrishnan et al., 2014; Wu et al., 2018) contains small organic molecules with up to 9 heavy atoms drawn from {C, N, O, F}. We use the DeFoG fixed split (Qin et al., 2025; Vignac et al., 2023). The small graph size makes QM9 a test of local chemical consistency: correct valency, atom type distribution, and bond type assignment.

**ZINC250k.** ZINC250k (Sterling & Irwin, 2015) contains drug-like molecules with up to 38 heavy atoms and a larger atom vocabulary (9 atom types including S, P, Cl, Br, I) than QM9. We use the predefined validation index split (Roos et al., 2026). The increased graph size and vocabulary make ZINC250k a harder test of distributional fidelity.

**MOSES.** MOSES (Polykovskiy et al., 2020) provides about 1.9M drug-like molecules with standardized splits: the official `train.csv` for training, `test.csv` for validation, and `test_scaffolds.csv` for testing. MOSES includes a comprehensive evaluation suite covering validity, uniqueness, novelty, internal diversity, FCD, scaffold similarity, and medicinal chemistry filter pass rates. MOSES is the largest and most chemically diverse of the three molecular benchmarks.

**Tree.** Tree (Bergmeister et al., 2024; Qin et al., 2025) is a structural graph benchmark where each graph is a tree with 64 nodes. Validity requires both connectivity and acyclicity, which are global constraints that demand coordination across all edge decisions. This makes Tree a challenging test for one-step samplers, where a single evaluation must produce a globally consistent graph structure.

### A.8 Implementation Details

**Architecture.** The backbone follows the graph transformer architecture of DeFoG (Qin et al., 2025), with $L$ message-passing layers, each consisting of multi-head self-attention over nodes and edges followed by a feed-forward block with RMSNorm. The node hidden dimension is $d_X = 256$, the edge hidden dimension is $d_E = 64$, and the global graph embedding has dimension $d_y = 64$, with 8 attention heads. The feed-forward dimensions are 256 for nodes and 128 for edges and the graph

Table 5: Hyperparameters per dataset.

| | QM9 | ZINC250k | MOSES | Tree |
|---|---|---|---|---|
| $\beta_{\text{start}}$ | 3 | 3 | 3 | 3 |
| $\beta_{\text{end}}$ | 12 | 24 | 20 | 32 |
| Prior variance $\sigma_0^2$ | 1 | 1 | 1 | 1 |
| Layers $L$ | 8 | 11 | 11 | 9 |
| Batch size | 2048 | 256 | 512 | 64 |
| Learning rate | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ |
| LR schedule | Cosine annealing | | | |
| Optimizer | AdamW (weight decay $10^{-12}$) | | | |
| EMA decay | 0.999 | 0.999 | 0.999 | 0.999 |
| Gradient clipping | 1.0 | 1.0 | 1.0 | 1.0 |
| Max epochs | 5000 | 500 | 300 | 100,000 |
| Edge weight $\lambda_E$ | 5.0 | 5.0 | 5.0 | 5.0 |
| $\lambda_{\text{BCD}}$, $\lambda_{\text{BTD}}$ | 1 | 1 | 1 | 1 |
| $\lambda_{\text{val}}$ | 1 | 2 | 1 | – |
| Heun substeps $d$ | 4 | 4 | 4 | 4 |
| Eval. samples | 10,000 | 10,000 | 25,000 | $5 \times 40$ |

embedding. Time embeddings for $s$ and $t$ are injected via sinusoidal encoding summed into the node and edge representations. The output head applies a per-block MLP followed by a softmax to produce interval-average probabilities $\boldsymbol{x}_{s,t}^{\theta}$ for each node-type and edge-type block. The terminal head $g_\psi$ is a single-layer slim variant (192 node / 48 edge hidden dimensions) that reads the concatenation of the last-layer hidden features and the current logits, as described in Appendix A.4.

**Precision schedule and prior.** We use the exponential cumulative precision schedule

$$\beta(t) = \beta_{\text{start}} \left[ \exp\left( t \log \frac{\beta_{\text{end}}}{\beta_{\text{start}}} \right) - 1 \right], \qquad t \in [0, 1], \tag{69}$$

with $\beta_{\text{end}} > \beta_{\text{start}} > 0$. This parameterization satisfies $\beta(0) = 0$ and $\beta(1) = \beta_{\text{end}} - \beta_{\text{start}}$. Here $\beta_{\text{start}}$ and $\beta_{\text{end}}$ are schedule parameters, not the initial and final values of $\beta(t)$. The prior is $\boldsymbol{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 I)$, with $\sigma_0^2 = 1$ for all datasets, independently of the precision schedule. Dataset-specific schedule parameters are listed in Table 5.

**Hyperparameters.** Table 5 summarizes the key hyperparameters for each dataset. The number of layers $L$ is the only architectural parameter that varies across datasets.

**Training.** All models are trained with AdamW and a cosine annealing learning rate schedule without warm-up. We maintain an exponential moving average (EMA) of the model parameters and use the EMA weights for all evaluations. Gradient norms are clipped to 1.0. Training is performed on NVIDIA RTX 3090 (24 GB) GPUs with distributed data parallelism.

**Node count sampling.** During generation, the number of nodes $n$ for each graph is sampled from the empirical node-count distribution of the training set. Specifically, we fit a categorical distribution over the observed node counts and sample $n$ independently for each generated graph.

**Evaluation protocol.** For QM9 and ZINC250k, we generate 10,000 molecules per evaluation run, consistent with the protocol of Roos et al. (2026). For MOSES, we generate 25,000 molecules following the protocol of Qin et al. (2025). For Tree, we generate 5 folds of 40 graphs each (200 total) and average the metrics across folds, following the protocol of Qin et al. (2025). All reported metrics are computed on these generated sets against the respective test splits. We use the MOSES evaluation framework (Polykovskiy et al., 2020) for MOSES metrics and the evaluation code of Qin et al. (2025) for structural graph metrics.

**CFM baseline.** We run Categorical Flow Maps (Roos et al., 2026) using the original authors' code with their published hyperparameters. CFM offers two loss variants, CSD and ECLD. We select the variant with better overall performance on each dataset: ECLD for the molecular benchmarks (QM9, ZINC250k, and MOSES) and CSD for Tree. CFM and BFM use the same data splits, graph representation, and evaluation protocol.

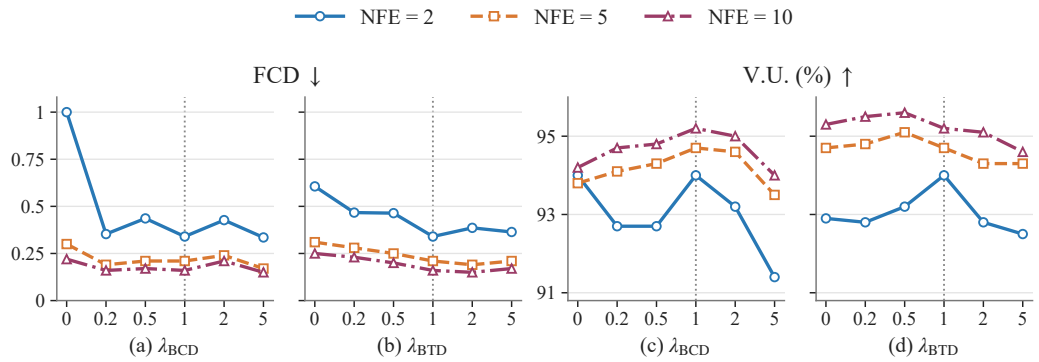


Figure 4: **Multi-step loss-weight sensitivity on QM9.** Panels **(a,b)** show FCD and **(c,d)** show V.U., with a shared scale per pair. Each sweep fixes the other weight at 1. Color, marker shape, and line style distinguish NFE $= 2, 5, 10$. Dotted lines mark the default weight of 1. Horizontal positions denote equally spaced tested settings.

Table 6: **BCD substep sensitivity on QM9.** Bold marks the best value per column.

| | 1-step | | | 2-step | | |
|---|---|---|---|---|---|---|
| **Substeps** $d$ | Valid ↑ | V.U. ↑ | FCD ↓ | Valid ↑ | V.U. ↑ | FCD ↓ |
| 2 | 95.9 | 92.6 | 0.56 | 96.8 | 93.1 | 0.51 |
| 4 (default) | **96.4** | 93.3 | **0.39** | **97.1** | **94.0** | 0.34 |
| 6 | 95.8 | 92.8 | 0.43 | 96.7 | 93.6 | 0.37 |
| 8 | 96.2 | **93.4** | 0.40 | 96.6 | 93.3 | 0.35 |
| 10 | 95.6 | 92.8 | 0.43 | 96.5 | 93.3 | **0.33** |

# B Additional Experiments

## B.1 Loss Weight Sensitivity

Figure 4 compares the 2-, 5-, and 10-step results for the weight sweeps in §3.3. At higher step counts the spread in FCD across settings narrows, consistent with each step covering a shorter part of the belief trajectory.

## B.2 BCD Substep Sensitivity

Table 6 varies the number of Heun substeps $d$ in BCD while keeping all other hyperparameters fixed. The default $d = 4$ offers the best one-step validity and FCD.

## B.3 Inference Cost

Table 7 reports the sampling wall-clock time per generated graph for the budgets used in the main Tables. All methods are timed on the same GPU with the same batch size per dataset, and the timing includes every step from prior sampling to discrete node and edge labels. One-step BFM is 6–9 times faster than 10-step GraphBSI and 44–55 times faster than 50-step DeFoG, with the speedup growing proportionally to the baseline's NFE. BFM and CFM have comparable per-evaluation cost, and the BFM terminal head is a single lightweight layer that adds negligible overhead. The speed advantage over iterative samplers therefore comes entirely from reducing the number of backbone evaluations.

## B.4 Terminal Head Ablation

Table 8 compares three decoding strategies at one and two transport steps on QM9: (i) direct argmax on the final belief logits, (ii) the lightweight terminal head (Appendix A.4), and (iii) an extra

Table 7: Sampling wall-clock time per generated graph (ms; median over repeated batches) on one NVIDIA RTX 3090 in float32 with EMA weights. All methods use the same batch size per dataset (256 for QM9 and ZINC250k, 128 for MOSES, 40 for Tree). Timing covers node-count sampling, prior initialization, all network evaluations and sampler updates, and decoding to discrete node and edge labels; it excludes RDKit/NetworkX conversion and metric computation. BFM and GraphBSI times include the terminal head. BFM, GraphBSI, and CFM use the architectures of the models in the main Tables; DeFoG uses its published checkpoints. "–" denotes a budget not used for that dataset in the main Tables.

| **Method** | **NFE** | **QM9** | **ZINC250k** | **MOSES** | **Tree** |
|---|---|---|---|---|---|
| BFM (ours) | 1 | 0.145 | 1.47 | 0.974 | 3.73 |
| BFM (ours) | 2 | 0.234 | 2.76 | 1.81 | 7.01 |
| CFM | 1 | 0.197 | 1.65 | 0.924 | 3.84 |
| CFM | 2 | 0.300 | 3.22 | 1.74 | 7.60 |
| GraphBSI | 10 | 0.940 | 13.1 | 8.48 | 32.8 |
| GraphBSI | 50 | 4.46 | 65.1 | 41.9 | 163 |
| DeFoG | 50 | 6.47 | 78.6 | 46.4 | 206 |
| DeFoG | 500 | 64.4 | 786 | 465 | – |
| DeFoG | 1000 | – | – | – | 4119 |
| DeFoG (50) / BFM (1) | | $44.5\times$ | $53.6\times$ | $47.6\times$ | $55.2\times$ |

Table 8: **Terminal head ablation on QM9.** NFE counts full backbone forward passes: transport steps plus the decoder's extra NFE. The terminal head reuses hidden features from the last transport step and adds no NFE. Bold marks the best value at each NFE budget.

| **NFE** | **Steps** | **Terminal decoding** | Valid ↑ | V.U. ↑ | FCD ↓ |
|---|---|---|---|---|---|
| 1 | 1 | Belief argmax | 70.8 | 68.8 | 1.14 |
| | 1 | Terminal head | **96.4** | **93.3** | **0.39** |
| 2 | 2 | Belief argmax | 69.7 | 68.1 | 0.93 |
| | 2 | Terminal head | 97.1 | 94.0 | 0.34 |
| | 1 | + extra reconstructor | **97.8** | **94.5** | **0.29** |
| 3 | 2 | + extra reconstructor | 97.3 | 94.6 | 0.18 |

reconstructor evaluation at the transported endpoint. NFE counts full backbone forward passes: the terminal head reuses hidden features from the last transport step and adds no extra NFE, while the extra reconstructor requires one additional forward pass.

Direct argmax produces poor validity, confirming that the final belief alone does not collapse onto a single category and explicit terminal decoding is necessary. BSI's monotonically increasing precision means that the transported endpoint belief, while not fully collapsed, already carries high information content about the true graph (Lienen et al., 2026; Petersen et al., 2026). The reconstructor, trained to extract the posterior mean at any precision level, can read this information and output accurate category probabilities. The terminal head distills this ability into a single lightweight layer that reads the backbone's hidden features $\boldsymbol{H}$ directly, resolving the remaining inter-category ambiguity from the structural correlations already encoded in $\boldsymbol{H}$. The terminal head recovers most of the quality gap without an additional backbone evaluation: at one step it raises validity from $70.8\%$ to $96.4\%$ and lowers FCD from $1.14$ to $0.39$. An extra reconstructor further improves FCD but requires doubling the NFE budget from 1 to 2, which is incompatible with the one-step generation setting.

**Comparison with CFM.** To verify that the terminal head's effectiveness is specific to BSI's belief framework rather than a generic benefit of an extra decoding step, we apply the same extra-reconstructor strategy to CFM on QM9. Table 9 shows that adding an extra decode evaluation to CFM yields negligible changes: at one step, validity increases by less than one point while FCD slightly worsens, and at two steps both metrics are essentially unchanged. In contrast, BFM's terminal head raises one-step validity from $70.8\%$ to $96.4\%$ and lowers FCD from $1.14$ to $0.39$ in Table 8. This disparity arises because BSI's reconstructor approximates the posterior mean $\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}_t]$, giving

Table 9: **Extra reconstructor evaluation on CFM (QM9).** Adding an extra decode step to CFM yields negligible improvement, in contrast to the large gains from BFM's terminal head in Table 8.

| NFE | Steps | Terminal decoding | Valid ↑ | V.U. ↑ | FCD ↓ |
|---|---|---|---|---|---|
| 1 | 1 | Direct output | 95.2 | 83.7 | 2.42 |
| 2 | 1 | + extra reconstructor | 96.1 | 83.5 | 2.52 |
| | 2 | Direct output | 96.0 | 93.2 | 0.60 |
| 3 | 2 | + extra reconstructor | 96.7 | 93.7 | 0.61 |

Table 10: **Valence weight sensitivity on QM9.** Bold marks the best value per column.

| | 1-step | | | 2-step | | |
|---|---|---|---|---|---|---|
| $\lambda_{\text{val}}$ | Valid ↑ | V.U. ↑ | FCD ↓ | Valid ↑ | V.U. ↑ | FCD ↓ |
| 0 | 95.1 | 92.3 | 0.41 | 96.2 | 93.3 | 0.34 |
| 0.2 | 95.0 | 92.4 | 0.42 | 96.0 | 92.8 | 0.35 |
| 0.5 | 95.3 | 92.5 | 0.43 | 96.5 | 93.2 | **0.34** |
| 1 (default) | 96.4 | 93.3 | **0.39** | **97.1** | **94.0** | **0.34** |
| 2 | 96.2 | 93.1 | 0.50 | 96.9 | 93.2 | 0.40 |
| 5 | **97.8** | **94.2** | 0.62 | 96.8 | 92.7 | 0.48 |

it access to the structural correlations encoded in the backbone's hidden features, as discussed in the Motivation paragraph of Appendix A.4. CFM does not share this posterior-mean structure, so an additional decode step provides little new information beyond what the transport output already contains.

### B.5 BFM Sampler and Euler Sampler

The same network provides both the interval-average predictor $\boldsymbol{x}^{\theta}_{s,t}$ and, at equal times, the local reconstructor $\boldsymbol{x}^{\theta}(\boldsymbol{z}_t, t)$. To test whether low-NFE quality comes from the learned finite-time map rather than from an accurate local field alone, we sample each BFM checkpoint in two ways: the BFM sampler (Eq. 22), or an Euler sampler that integrates the normalized belief ODE (Eq. 13) using the same network's local reconstructor. Both samplers use the same inference grid, initial beliefs, node counts, and terminal head, and NFE ranges from 1 to 64. Sample sizes and metrics follow the main tables. Because these are separate sampling runs, values at NFE $\leq 2$ can differ slightly from the main tables.

Figure 5 shows that the Euler sampler collapses at one step across all four datasets, whereas the BFM sampler retains substantial sample quality. The gap narrows steadily as NFE increases, and the two samplers converge to comparable quality at 32–64 evaluations. This suggests that the network learns an accurate local field. The one- and two-step quality gains are therefore attributable to the learned finite-time map, not to a better local reconstructor.

### B.6 Sensitivity to the Inference Grid

For $K > 1$ BFM sampling steps, the inference grid in Eq. 22 places the transport nodes at $\tau_{t_k} = (k/K)^{\rho}\,\tau_1$, where $\tau$ is the signal-to-noise time of Eq. 12. All results in the main text use $\rho = 1$, i.e. uniform steps in $\tau$. $\rho < 1$ allocates shorter steps near the end of generation and $\rho > 1$ near the start. We vary $\rho \in \{0.5, 0.75, 1, 1.5, 2\}$ for $K \in \{2, 4, 8\}$ without retraining; results are in Figure 6.

Front-loading steps near the start ($\rho \geq 1.5$) degrades both sample quality and distributional fidelity, most strongly at two steps. Shifting steps towards the end ($\rho < 1$) can improve two-step sample quality on several datasets but does not consistently lower FCD. Differences shrink as $K$ grows, and no alternative grid is uniformly better than the default $\rho = 1$, which we use throughout.

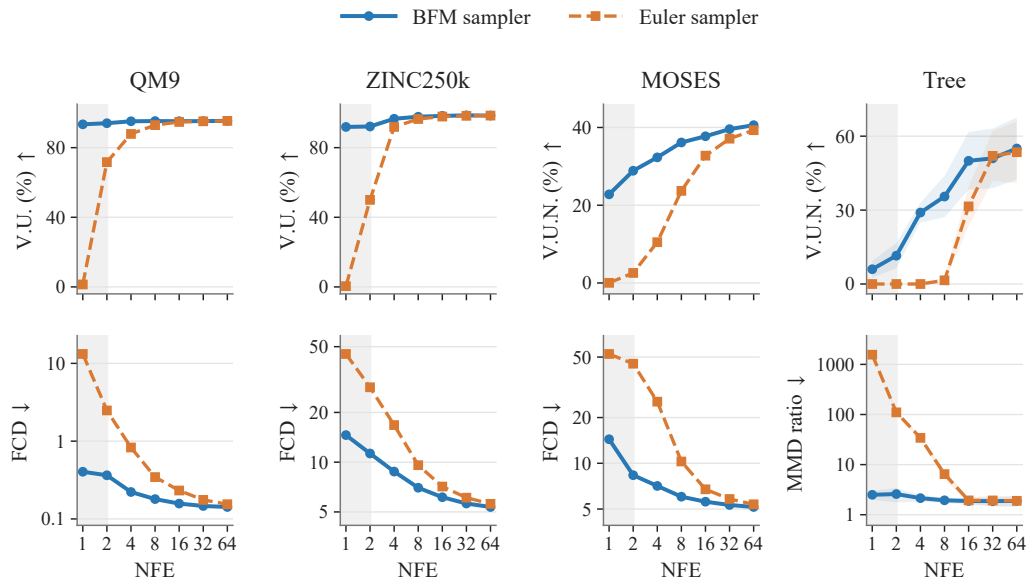


Figure 5: BFM sampler versus Euler sampler as a function of NFE. Both use the same trained network, inference grid, initial beliefs, node counts, and terminal head. Top: V.U. (QM9, ZINC250k) or V.U.N. (MOSES, Tree). Bottom: FCD or average MMD ratio (log scale). Gray shading marks the one- and two-step budgets of the main tables. Tree bands show the standard deviation over five folds.

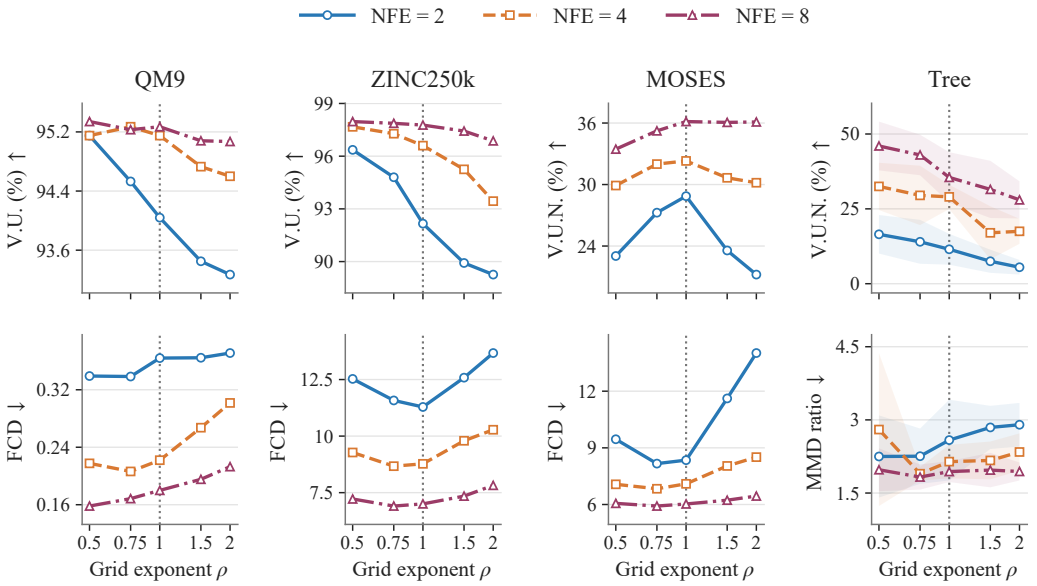


Figure 6: Sensitivity of multi-step BFM sampling to the grid exponent $\rho$ in $\tau_{t_k} = (k/K)^{\rho}\tau_1$, for $K \in \{2, 4, 8\}$ evaluations. Top: V.U. (QM9, ZINC250k) or V.U.N. (MOSES, Tree). Bottom: FCD or average MMD ratio. The dotted line marks the default $\rho = 1$ used in the main text. Tree bands show the standard deviation over five folds.

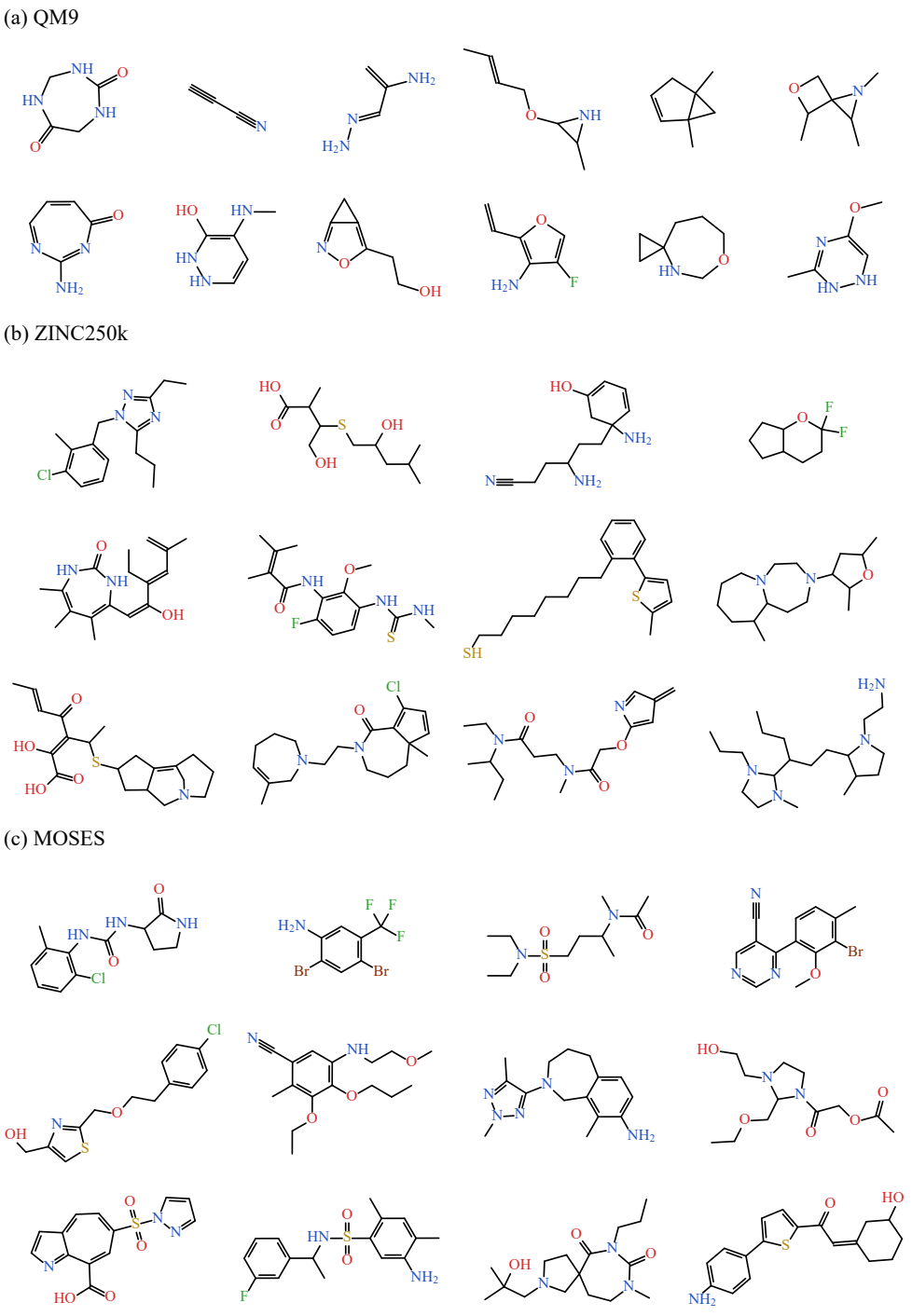


Figure 7: Selected one-step (NFE $= 1$) molecular samples from BFM on (a) QM9, (b) ZINC250k, and (c) MOSES, with 12 samples per dataset. Selection criteria are described in Appendix B.8.

### B.7 Valence Weight Sensitivity

Table 10 varies the valence penalty weight $\lambda_{\text{val}}$ on QM9. Even without the valence penalty ($\lambda_{\text{val}} = 0$), BFM already achieves $95.1\%$ one-step validity and $0.41$ FCD, indicating that the belief flow training and terminal head alone produce chemically plausible molecules. Adding a moderate penalty further

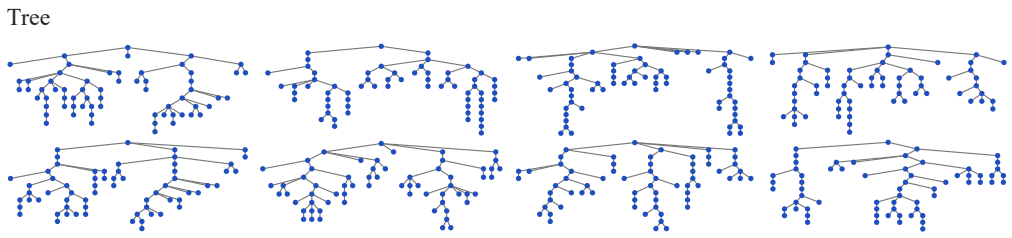


Figure 8: Selected one-step (NFE $= 1$) Tree samples from BFM: 8 of the 12 valid trees among 200 generated graphs. Selection criteria are described in Appendix B.8.

improves validity with only a slight increase in FCD. Larger weights push one-step validity even higher but shift the training signal away from distributional fidelity, raising FCD. The default $\lambda_{\text{val}} = 1$ balances validity and FCD.

### B.8 Qualitative Samples

Figures 7 and 8 show one-step (NFE $= 1$) BFM samples. For molecules, we show valid, connected molecules that occur once in the generated set, are novel with respect to the training set, and carry no formal charges. For ZINC250k and MOSES we additionally require passing the MOSES filters and exclude three- and four-membered rings. Among these, molecules are chosen for structural diversity across size terciles using MaxMin selection on Morgan fingerprints. For Tree, we show 8 of the 12 valid trees among the 200 one-step samples.