KLEIN FLOW MATCHING IN HYPERBOLIC LATENT SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating realistic graphs requires capturing both local connectivity and large-scale organization, including hierarchical and tree-like patterns. Hyperbolic space is naturally suited to representing these patterns, making it a promising latent geometry for graph generation. We introduce Klein Flow Matching (KFM), a two-stage latent graph generation model that exploits a simple geometric property: hyperbolic geodesics appear as straight line segments in Klein coordinates. Specifically, KFM represents each conditional velocity using a fixed direction and a time-dependent scalar factor. This gives explicit positions and velocity targets for simulation-free flow-matching training. A graph autoencoder provides deterministic latent targets and condition codes; a conditional transformer learns their transport from a wrapped Gaussian base using the Klein metric. Sampling combines Riemannian Heun integration with graph decoding, including structural constraints. Across five main benchmarks, KFM shows competitive results in degree and clustering statistics. On additional benchmarks for generating planar graphs and trees, constrained decoding and candidate selection yield 100% jointly valid, unique, and novel outputs, with the lowest aggregate structural-discrepancy ratio among the compared models on trees. These results demonstrate the practical promise of KFM for generating graphs across diverse structural regimes.

1 INTRODUCTION

Generating realistic graphs requires reproducing both local connectivity and larger-scale structures that distinguish graph distributions. Autoregressive models such as GraphRNN (You et al., 2018) learn these dependencies through sequential node and edge generation, whereas discrete diffusion models such as DiGress (Vignac et al., 2023) learn to reverse successive perturbations of graph structure. Latent graph generators take a different approach by separating graph reconstruction from distribution learning, where graphs are encoded into latent representations, a generative model learns their distribution, and a decoder maps sampled representations back to graphs. The choice of latent geometry is thus an important inductive bias for representing graph structure and determines the geometric operations available for learning and sampling the latent distribution.

For graphs with hierarchical organization, hyperbolic space provides a natural choice of latent geometry: its exponential volume growth accommodates tree-like expansion, while hyperbolic embeddings can capture both hierarchy and similarity (Nickel & Kiela, 2017). Exploiting this geometry for generation requires a model that can learn distributions within it. HypDiff (Fu et al., 2024) models graph latents through a diffusion process constrained by radial and angular geometry. Riemannian flow matching (Chen & Lipman, 2024) extends flow matching (Lipman et al., 2023) to manifolds by regressing velocity fields along prescribed probability paths. GGBall (Bu et al., 2026) applies this framework to graph generation by combining a hyperbolic vector-quantized autoencoder with a Poincaré flow prior based on closed-form geodesics.

For hyperbolic latent flows, the coordinate model determines not only how latent points are represented but also the form of the velocity targets used to train the generative field. The Poincaré and Klein models describe the same intrinsic geometry, but their geodesics have different coordinate representations (Mao et al., 2026). Poincaré geodesics generally follow circular arcs, so their coordinate velocity changes direction along the path. In Klein coordinates, the same intrinsic geodesics follow straight chords: for each pair of distinct endpoints, the direction remains fixed and

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.

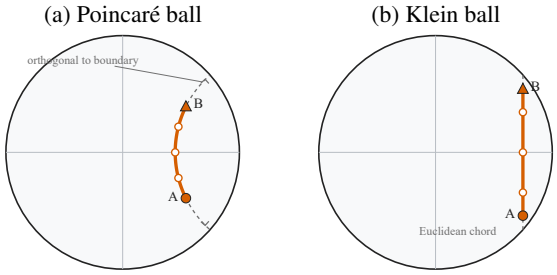

Figure 1: Geometric intuition behind Klein Flow Matching: the velocity has fixed direction and variable coordinate speed. The same hyperbolic geodesic appears as a Poincaré arc (left) and a Klein chord (right). Markers indicate equal time increments at constant hyperbolic speed. KFM exploits the chord's fixed direction and time-dependent coordinate speed to construct conditional velocity targets. A Lorentz projection construction is presented later in Figure 3.

all time dependence is captured by a scalar speed. Figure 1 illustrates this distinction. This separates the endpoint displacement from its time dependence, providing a simple representation of velocity supervision without changing the underlying hyperbolic geometry.

We turn this motivation into *Klein Flow Matching* (KFM), a two-stage model for graph generation using hyperbolic latent transport. Building on the closed-form Klein geodesics (Nielsen, 2021), we express the endpoint-conditioned velocity as $\dot{x}_t = \eta_t(x_1 - x_0)$, where η_t is an explicit scalar determined by the endpoints x_0, x_1, and time. These expressions provide training positions and velocity targets directly.

KFM first trains a graph autoencoder. The encoder supplies deterministic hyperbolic latent targets and structural condition codes. After standardizing the targets in the tangent space at the origin, a conditional transformer learns a vector field that transports a wrapped Gaussian base distribution toward the graph-latent distribution. Velocity errors are measured with the Klein metric, so regression accounts for their hyperbolic rather than Euclidean length. At generation time, a Riemannian Heun solver integrates the learned field using exponential-map updates and exact parallel transport. The resulting latent is then mapped back through inverse standardization and decoded into a graph.

Our contributions are summarized as follows:

- We formulate endpoint-conditioned hyperbolic flow matching using explicit geodesic paths and velocity targets that separate a fixed chord direction from a time-dependent scalar.
- We integrate this supervision into a two-stage graph generator, combining structural conditioning, Klein-metric velocity regression, and Riemannian Heun integration.
- We demonstrate strong degree and clustering matching on synthetic and real-world benchmarks, and characterize performance differences across various backbones.

2 Related Work

Graph Generation. Graph generators seek to capture structural and topological patterns within graphs. Autoregressive models such as GraphRNN (You et al., 2018) learn such patterns among nodes and edges in a sequential manner. Graph-space diffusion methods, including GDSS (Jo et al., 2022), GraphGDP (Huang et al., 2022), DiGress (Vignac et al., 2023), and EDGE (Chen et al., 2023), instead generate graphs by reversing successive perturbations of graph states. Latent-variable models take a different approach by learning representations from which graphs can be reconstructed. GraphVAE (Simonovsky & Komodakis, 2018) decodes continuous latent variables into graphs, while Latent Graph Diffusion (LGD) (Zhou et al., 2024) learns a diffusion model over representations supplied by a pretrained graph encoder. GLAD (Nguyen et al., 2025) instead uses quantized latent representations and learns their prior through diffusion bridges. These approaches separate the representation of graph structure from the mechanism used to generate latent samples and provide a broad context for the hyperbolic generators discussed below.

Flow Matching for Graphs. Flow matching (Lipman et al., 2023) learns generative dynamics by regressing vector fields along prescribed probability paths. For graph generation, CatFlow (Eijkelboom et al., 2024) develops a flow-matching method for categorical data through a variational formulation. DeFoG (Qin et al., 2025), GGFlow (Hou et al., 2025), and SimGFM (Luo et al., 2026) develop discrete flow-matching formulations for graph generation, studying how probability paths and sampling procedures can accommodate graph-valued data. Taking a geometric approach to path construction, BWFlow (Jiang et al., 2026) represents graphs through Markov random fields and constructs interpolation paths using Bures-Wasserstein transport.

Hyperbolic Graph Generation. Hyperbolic representations provide an inductive bias for hierarchical and tree-like relational data (Nickel & Kiela, 2017; Chami et al., 2019). HAEGAN (Qu & Zou, 2024) combines a hyperbolic autoencoder with a Wasserstein GAN in the Lorentz model, learning to generate latent representations that are decoded into trees and molecular graphs. Diffusion-based approaches include HGDM (Wen et al., 2024), which jointly models hyperbolic node representations and Euclidean adjacency matrices, and HypDiff (Fu et al., 2024), which incorporates this geometry into a latent diffusion model for graph generation. GGBall (Bu et al., 2026) adopts Riemannian flow matching, combining a hyperbolic vector-quantized autoencoder with a continuous flow prior in the Poincaré ball. Compared with GGBall, KFM uses unquantized graph-level targets and formulates the flow in Klein coordinates, exposing the direction-speed separation of its conditional velocity targets.

Riemannian Flow Matching. Riemannian flow matching (Chen & Lipman, 2024) extends flow matching to manifolds through geometry-dependent conditional paths. For instance, FoldFlow (Bose et al., 2024) and ReQFlow (Yue et al., 2025) exploit rigid-motion and quaternion representations for protein generation, FlowMM (Miller et al., 2024) models periodic crystal structures, and Fisher-Flow uses Fisher-Rao geometry for categorical data (Davis et al., 2024). Broader developments include learning interpolants under data-induced metrics (Kapuśniak et al., 2024) and modeling families of distributions through Wasserstein geodesics (Haviv et al., 2025). Other approaches exploit geometric representations: DiffeoCFM (Collas et al., 2025) reduces flow matching under suitable Euclidean pullback metrics to Euclidean flow matching, while Cartan flow matching (Ruscelli et al., 2026) uses symmetric-space structure to reformulate the problem in a subspace of the isometry group's Lie algebra. KFM focuses on the coordinate representation of hyperbolic supervision: closed-form Klein geodesics (Nielsen, 2021; Mao et al., 2026) express each conditional velocity through a fixed endpoint displacement and a time-dependent scalar, while the regression loss retains the Klein metric.

3 METHOD

Klein Flow Matching (KFM) is a two-stage graph generator. We first train a graph autoencoder, then fit a conditional latent flow using explicit Klein geodesic velocities. Sampling integrates the field from noise and decodes the resulting latent.

3.1 LATENT GRAPH GENERATION

Let $\{G_i\}_{i=1}^{M}$ be undirected, loop-free training graphs with at most $N_{\max}$ nodes. The encoder produces a Gaussian posterior over origin-tangent vectors $u \in \mathbb{R}^d$, $q_\phi(u \mid G) = \mathcal{N}(\mu_G, \mathrm{diag}(\sigma_G^2))$, and a deterministic graph-level condition $c_G \in \mathbb{R}^{d_c}$. The condition c_G is learned from the graph representation. For Grid, Planar, and Tree, we augment it with an embedding of graph statistics, such as size and degree distribution.

Autoencoder training maps posterior samples to the Klein ball using the origin exponential map, $z = \exp_0^K(u)$. The decoder $D_\psi(z, c)$ uses $\log_0^K(z)$ and c to predict node presence, edge probabilities, and auxiliary graph statistics. Masked reconstruction supervises nodes and edges, auxiliary losses supervise graph statistics, and a dimension-normalized Gaussian KL penalty regularizes the posterior toward a standard Gaussian (Kingma & Welling, 2014). During flow learning, the autoencoder parameters remain fixed, and posterior means define deterministic targets $z_G^\star = \exp_0^K(\mu_G)$. Both the flow and decoder are conditioned on c_G. Appendix C specifies the encoder and decoder architectures and the autoencoder training objective.

3.2 Geodesic Transport in Klein Coordinates

Geometry and distributions. The unit Klein ball $\mathbb{K}^d = \{x \in \mathbb{R}^d : \|x\|_2 < 1\}$ represents hyperbolic space of curvature -1 (Mao et al., 2026). Its metric and tangent norm are

$$g_x^K = \frac{I}{1-\|x\|_2^2} + \frac{xx^\top}{(1-\|x\|_2^2)^2}, \qquad \|v\|_{K,x}^2 = \frac{\|v\|_2^2}{1-\|x\|_2^2} + \frac{(x^\top v)^2}{(1-\|x\|_2^2)^2}. \tag{1}$$

At the origin, $\exp_0^K(u) = \tanh(\|u\|_2)u/\|u\|_2$ and $\log_0^K(x) = \operatorname{artanh}(\|x\|_2)x/\|x\|_2$, with their continuous values at zero. Before flow training, we center and rescale origin-tangent coordinates, then map back to the ball. Writing $z_i^\star = z_{G_i}^\star$, let m and s be the elementwise training mean and standard deviation of $\log_0^K(z_i^\star)$, with each component of s floored at 10^{-4}. The standardized target x_1 and independent noise endpoint x_0 are

$$T(z) = \exp_0^K\Big((\log_0^K(z) - m) \oslash s\Big), \qquad x_1 = T(z_G^\star), \qquad x_0 = \exp_0^K(\epsilon), \quad \epsilon \sim \mathcal{N}(0, \sigma_0^2 I). \tag{2}$$

Here $\sigma_0 > 0$ is the base scale and $\oslash$ denotes elementwise division. The transformation T is invertible in exact arithmetic but is not generally an isometry; the flow uses the Klein metric after standardization. The wrapped Gaussian base $p_0 = (\exp_0^K)_\# \mathcal{N}(0, \sigma_0^2 I)$ requires no density evaluation. To maintain a margin from the boundary, base samples and targets are projected to a smaller ball. This projection changes their distributions and prevents exact inversion when clipping occurs. Appendix B.2 defines the projection, and Appendix D.1 quantifies its frequency and effects on reconstruction.

An explicit path and its velocity. Geodesics connecting noise and target endpoints supply training positions and velocities. In Klein coordinates, they follow Euclidean chords, with generally nonlinear time parameterization for constant hyperbolic speed. For distinct endpoints x_0, x_1, set

$$\begin{aligned} &\Delta = x_1 - x_0, \qquad a = 1 - \|x_0\|_2^2, \qquad b = x_0^\top \Delta, \qquad S = \sqrt{a\|\Delta\|_2^2 + b^2}, \\ &\rho = d_K(x_0, x_1) = \operatorname{arcosh}\left(\frac{1 - x_0^\top x_1}{\sqrt{(1-\|x_0\|_2^2)(1-\|x_1\|_2^2)}}\right). \end{aligned} \tag{3}$$

Writing $B_t = S\cosh(t\rho) + b\sinh(t\rho)$, the geodesic and its coordinate velocity are

$$x_t = x_0 + \frac{a\sinh(t\rho)}{B_t}\Delta, \qquad \dot{x}_t = \frac{aS\rho}{B_t^2}\Delta, \qquad 0 \le t \le 1. \tag{4}$$

Equation (4) parameterizes $\exp_{x_0}^K(t \log_{x_0}^K(x_1))$ (Nielsen, 2021). Since $S > |b|$, $B_t > 0$; the path interpolates the endpoints inside the ball with constant hyperbolic speed $\|\dot{x}_t\|_{K,x_t} = \rho$. For coincident endpoints, set $x_t = x_0$ and $\dot{x}_t = 0$. The coordinate velocity has fixed direction Δ and a time-dependent scalar factor. These expressions give training pairs without ODE integration. Appendix A.2 derives these path and velocity expressions.

3.3 Conditional Klein Flow Matching

Write $c_i = c_{G_i}$. For the ideal construction before radius projection, let $\widehat{q}_{1,C} = M^{-1}\sum_i \delta_{(T(z_i^\star), c_i)}$ be the empirical joint distribution of targets and conditions. We sample $(x_1, c) \sim \widehat{q}_{1,C}$, independently draw $x_0 \sim p_0$ and $t \sim \mathcal{U}(0,1)$ in the theoretical objective, and construct $(x_t, \dot{x}_t)$ using Eq. (4). The velocity field is learned by regressing the path velocity from (x_t, t, c). Following Riemannian flow matching (Chen & Lipman, 2024), we minimize

$$\mathcal{L}_{\mathrm{KFM}}(\theta) = \mathbb{E}_{t,x_0,(x_1,c)}\left[\|v_\theta(x_t,t,c) - \dot{x}_t\|_{K,x_t}^2\right]. \tag{5}$$

The metric in Eq. (1) weights errors by hyperbolic length: its second term additionally penalizes the radial component, with increasing sensitivity near the boundary.

From conditional paths to a generative field. For fixed c and t, let $p_t(\cdot \mid c)$ be the law of x_t and define $\bar{u}(x,t,c) = \mathbb{E}[\dot{x}_t \mid x_t = x, c]$. The conditional expectation averages path velocities at the given state and condition. Conditioning the loss on (x_t, t, c) gives

$$\mathcal{L}_{\mathrm{KFM}}(\theta) = \mathbb{E}\|v_\theta(x_t,t,c) - \bar{u}(x_t,t,c)\|_{K,x_t}^2 + \Delta_{\mathrm{FM}}, \qquad \Delta_{\mathrm{FM}} \text{ independent of } \theta. \tag{6}$$

With finite Klein-norm second moments, the decomposition identifies $\bar{u}$ as the marginal regression target. Under the regularity conditions in Appendix B, it transports $p_t(\cdot \mid c)$ for $0 < t < 1$. The learned field needs no endpoint x_1 at generation time, and its trajectories need not be geodesics. The atomic empirical endpoint law is a limiting target for a smooth flow.

Conditional vector field. We parameterize v_θ with a conditional DiT (Peebles & Xie, 2023) using the geometric features $f_K(x) = [x \,\|\, \log_0^K(x) \,\|\, \|x\|_2 \,\|\, (1 - \|x\|_2^2)^{-1/2}] \in \mathbb{R}^{2d+2}$, where $\|$ inside the brackets denotes concatenation. These combine ball and origin-tangent coordinates with radial features. A learned linear projection forms four latent tokens; time and condition embeddings modulate the transformer blocks. The network outputs a coordinate tangent vector in $T_x\mathbb{K}^d \simeq \mathbb{R}^d$. Geometry enters through the paths, metric loss, and integration; the network is not assumed isometry-equivariant. Appendix C.5 details the token construction and the modulation of transformer blocks by time and condition embeddings.

We jointly train conditional and unconditional predictions by replacing c with the null code $0 \in \mathbb{R}^{d_c}$ with probability p_{drop} during training. At sampling time, classifier-free guidance (Ho & Salimans, 2022) uses

$$v_w(x, t, c) = v_\theta(x, t, 0) + w\big[v_\theta(x, t, c) - v_\theta(x, t, 0)\big]. \tag{7}$$

The two predictions share a tangent space. Setting $w = 1$ recovers the conditional field; $w \neq 1$ modifies the transport beyond the original probability-path identity.

3.4 Manifold Sampling and Graph Reconstruction

To generate a graph, independently sample $c \sim \widehat{p}_C = M^{-1} \sum_i \delta_{c_i}$ and $x_0 \sim p_0$, then integrate $\dot{x} = v_w(x, t, c)$ from $t = 0$ to $t = 1$, holding c fixed. For Grid, Planar, and Tree, the condition is paired with graph statistics from the same training graph; c may be perturbed before integration. Appendix C.6 specifies the associated graph-size constraints and condition perturbations. For K steps with $h = 1/K$ and $t_k = kh$, Riemannian Heun computes a predictor and corrected exponential-map update:

$$\begin{aligned} v_k &= v_w(x_k, t_k, c), \qquad \widetilde{x}_{k+1} = \exp_{x_k}^K(h v_k), \\ x_{k+1} &= \exp_{x_k}^K\left(\frac{h}{2}\left[v_k + P^K_{\widetilde{x}_{k+1} \to x_k} v_w(\widetilde{x}_{k+1}, t_k + h, c)\right]\right). \end{aligned} \tag{8}$$

The second velocity is evaluated at the predictor $\widetilde{x}_{k+1}$. Exact parallel transport P^K brings it back to $T_{x_k}\mathbb{K}^d$ before averaging with v_k. With a sufficiently smooth field, exact geometric operations, and inactive clipping on a compact interior region, the method has second-order global accuracy. Appendix B.2 derives this accuracy result and gives the exact parallel transport formula. Inverse standardization restores the decoder's latent coordinates; decoding uses the same condition:

$$z_{\text{gen}} = T^{-1}(x_K) = \exp_0^K\Big(s \odot \log_0^K(x_K) + m\Big), \qquad (p^V, p^E) = D_\psi(z_{\text{gen}}, c). \tag{9}$$

Here $\odot$ denotes elementwise multiplication, and p^V, p^E are node presence and edge probabilities. Without prescribed structural constraints, we select node slots and threshold edge probabilities. Grid uses edge-budget decoding; Planar and Tree use constrained decoding with candidate selection. Their constraints guarantee validity; uniqueness and novelty are measured after selection. Appendix C.6 specifies the decoding rules and candidate selection criteria.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate graph topology on five main benchmarks: Community (community structure), Ego (citation-network neighborhoods), Grid (two-dimensional lattices), MUTAG (nitro-compound molecular graphs), and IMDB-B (actor co-appearance networks) (You et al., 2018; Fu et al., 2024). Planar and Tree test constrained generation. Additional results on Community-small and Ego-small, including baseline comparisons and ablations, appear in Appendix D.

Baselines. The main comparison includes VGAE (Kipf & Welling, 2016), GraphRNN (You et al., 2018), graph diffusion models GDSS (Jo et al., 2022), DiGress (Vignac et al., 2023), GraphGDP (Huang et al., 2022), and EDGE (Chen et al., 2023), hyperbolic latent diffusion HypDiff (Fu et al., 2024), and discrete flow matching SimGFM (Luo et al., 2026). We quote the first seven baselines from Fu et al. (2024) and reproduce SimGFM with its original code under our data-processing and evaluation pipeline. All non-KFM Planar/Tree results are quoted from Luo et al. (2026). Quoted results retain their source protocols. We also reproduce GGBall (Bu et al., 2026) on the five main datasets using the same data splits, preprocessing, and evaluation protocol as KFM. Appendix D.5 reports its five-seed means and configurations separately from the published small-dataset results.

Evaluation protocol. Following prior graph-generation protocols (You et al., 2018; Fu et al., 2024), we compare degree, clustering, and normalized-Laplacian spectral distributions using conventional MMD (Gretton et al., 2012). Lower values indicate closer distribution matching. KFM results are means over five independent training seeds. On the five main datasets, each reference graph is matched to the generated graph closest in node count, permitting repeated selections. For Planar/Tree, V.U.N. measures the joint fraction of valid, unique, and training-novel outputs; Ratio averages five statistic distances normalized by train–test distances (Luo et al., 2026). Appendix C.7 specifies the estimators and protocols.

Implementation details. KFM runs on a single NVIDIA GeForce RTX 4090 GPU. Both stages use AdamW with cosine learning-rate annealing, weight decay 10^{-4}, and gradient-norm clipping at 1.0. Dataset configurations are selected using validation metrics. Checkpoint rules are specified separately in Appendix C.5. Appendix C gives architecture, training, and decoding details; Appendix C.8 lists the configurations.

4.2 Main Results

Tables 1 and 2 report the main and constrained-generation results. The latter's training-set row is a quoted reference, not a novelty evaluation against itself.

KFM performs best or ties for best on seven of the fifteen main-dataset metrics, all in degree or clustering; other methods have lower spectral MMD on each dataset (Table 1). On Planar and Tree, KFM achieves 100% V.U.N. with constrained decoding and candidate selection, which guarantee validity by construction. Its Tree Ratio of 1.2 is the lowest among the compared generators; the Planar Ratio of 13.8 remains above several baselines.

Figure 2 shows generated connectivity on Community, MUTAG, and Tree. KFM Tree samples use the same constrained procedure; SimGFM Tree examples use a separate sampling run and do not underlie the quoted metrics in Table 2.

4.3 Ablation Studies

We compare the full model with two variants. The backbone ablation replaces DiT with a U-Net of approximately matched parameter count, holding the data, encoder–decoder, conditioning, optimization, and sampling settings fixed. For the coordinate ablation, the learned Klein VAE and DiT are retained, while flow matching is performed in Poincaré coordinates via isometric Klein–Poincaré transformations before and after flow integration; no separate Poincaré autoencoder is trained. Results are averaged over five training seeds.

Table 3 shows that DiT has lower mean MMD than U-Net in 12 of 15 comparisons, ties on MUTAG clustering, and has higher means on Grid degree and spectral MMD. Klein has lower means than Poincaré in 13 of 15 comparisons; Poincaré has lower spectral MMD on Grid and MUTAG. Table 4 shows identical V.U.N. across variants, with Klein + DiT attaining the lowest mean Ratio on both datasets.

Contribution of the learned transport. We compare KFM with two no-flow controls using the frozen encoder–decoder. AE posterior samples $u \sim q_\phi(u \mid G_i)$ from a uniformly drawn training graph; VAE prior samples $u \sim \mathcal{N}(0, I)$. Both decode $D_\psi(\exp_0^K(u), c_i)$ using shared training-derived

Table 1: MMD on the five main datasets (↓). SimGFM is reproduced using its original code with our data-processing and evaluation pipeline; the remaining baselines in this table are quoted from Fu et al. (2024). Additional GGBall reproductions under the KFM evaluation protocol appear in Table 15. KFM reports five-seed means; sample standard deviations appear in Table 3. Our evaluator matches reference node counts with replacement. Clust./Spec. denote clustering/spectral statistics. **Bold**, underline, and *italics* mark the three best distinct displayed values; ties share a rank.

Method	Community			Ego			Grid		
	Degree	Clust.	Spec.	Degree	Clust.	Spec.	Degree	Clust.	Spec.
VGAE	0.365	0.025	0.507	0.414	0.156	0.456	**0.050**	2.000	*0.145*
GraphRNN	0.002	0.027	**0.004**	0.206	0.539	0.157	0.203	0.043	**0.042**
GDSS	0.094	0.031	0.052	1.034	0.143	0.667	*0.111*	0.005	0.886
DiGress	0.226	0.158	0.194	0.110	*0.056*	0.122	0.689	1.115	0.203
GraphGDP	0.046	*0.016*	0.042	*0.059*	0.115	*0.054*	0.872	1.001	0.174
EDGE	*0.021*	0.013	*0.040*	0.023	0.048	0.023	0.223	0.072	0.802
HypDiff	0.002	**0.010**	0.028	0.075	0.090	**0.015**	0.422	0.665	0.137
SimGFM	0.022	0.065	0.081	0.389	0.163	0.076	0.965	*0.024*	1.005
KFM (ours)	**0.001**	0.020	0.061	**0.011**	**0.019**	0.067	0.110	**0.001**	0.262

Method	MUTAG			IMDB-B		
	Degree	Clust.	Spec.	Degree	Clust.	Spec.
VGAE	0.255	2.000	0.744	0.514	1.405	0.700
GraphRNN	0.537	0.013	0.476	0.137	0.252	0.423
GDSS	0.074	*0.021*	**0.003**	0.904	1.729	0.748
DiGress	0.100	0.351	0.082	0.166	0.425	0.159
GraphGDP	0.127	0.057	*0.050*	0.123	0.638	0.257
EDGE	0.024	0.597	0.468	0.041	0.874	**0.026**
HypDiff	*0.048*	**0.001**	0.040	**0.034**	*0.257*	0.030
SimGFM	0.121	0.307	0.173	0.345	1.252	0.096
KFM (ours)	**0.001**	**0.001**	0.131	*0.093*	**0.027**	*0.074*

Table 2: Planar/Tree V.U.N. (%, ↑) and Ratio (↓). **Bold**/underline mark the best/second-best distinct values among generative models; ties share a rank. All non-KFM entries, including SimGFM, are quoted from Luo et al. (2026). KFM uses constrained decoding and candidate selection (Appendix C.6).

Model	Class	Planar		Tree	
		V.U.N.↑	Ratio↓	V.U.N.↑	Ratio↓
Train set	–	100	1.0	100	1.0
GraphRNN	Autoregressive	0.0	490.2	0.0	607.0
GRAN	Autoregressive	0.0	2.0	0.0	607.0
BiGG	Autoregressive	5.0	16.0	75.0	5.2
GraphGen	Autoregressive	7.5	210.3	95.0	33.2
AUTOGRAPH	Autoregressive	87.5	**1.5**	–	–
EDGE	Diffusion	0.0	431.4	0.0	850.7
BwR (EDP-GNN)	Diffusion	0.0	251.9	0.0	11.4
DiGress	Diffusion	77.5	5.1	90.0	1.6
HSpectral	Diffusion	95.0	2.1	**100.0**	4.0
GruM	Diffusion	90.0	1.8	–	–
DisCo	Diffusion	83.6	–	–	–
Cometh	Diffusion	92.5	–	–	–
Cometh-PC	Diffusion	99.5	–	–	–
CatFlow	Flow	80.0	–	–	–
DeFoG (50 steps)	Flow	95.0	3.2	73.5	2.5
DeFoG (1000 steps)	Flow	99.5	1.6	96.5	1.6
SimGFM (200 steps)	Flow	**100.0**	9.3	99.5	1.5
KFM (ours)	Flow	**100.0**	13.8	**100.0**	**1.2**

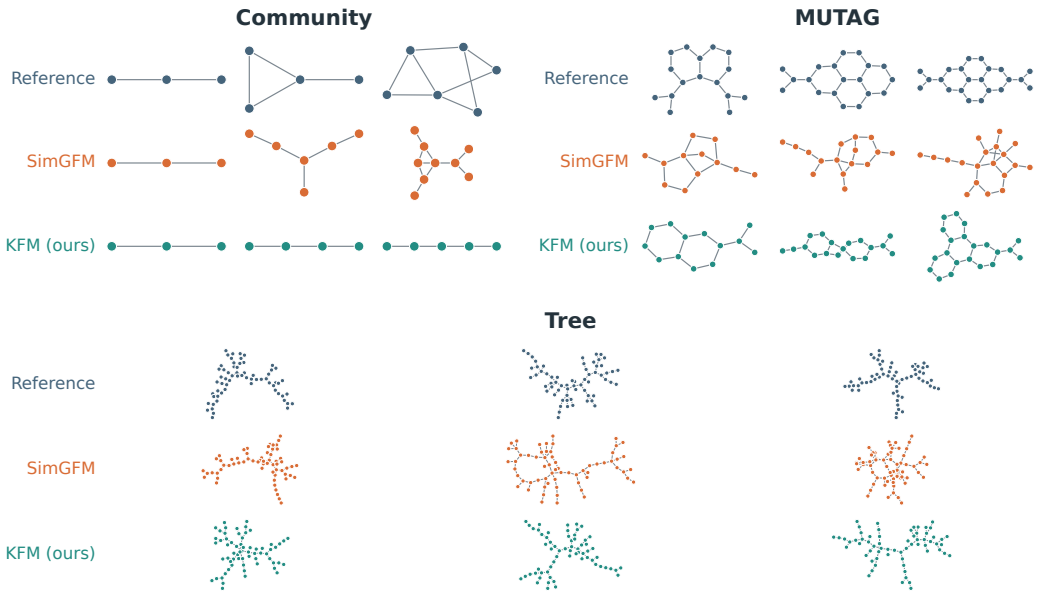

Figure 2: Graph examples on Community, MUTAG, and Tree. Rows show held-out reference graphs, SimGFM samples, and KFM samples. Community/MUTAG examples use node-count quantiles; Tree examples use fixed-seed random sampling.

Table 3: Main-dataset ablations: MMD (↓). Entries are mean ± sample standard deviation over five seeds. Bold marks the lowest displayed mean, including ties.

Dataset	Variant	Degree↓	Clustering↓	Spectral↓
Community	Klein + DiT	**0.001±0.001**	**0.020±0.010**	**0.061±0.007**
	Klein + U-Net	0.009±0.006	0.073±0.027	0.085±0.010
	Poincaré + DiT	0.005±0.004	0.055±0.035	0.101±0.032
Ego	Klein + DiT	**0.011±0.005**	**0.019±0.013**	**0.067±0.002**
	Klein + U-Net	0.037±0.020	0.059±0.062	0.072±0.002
	Poincaré + DiT	0.118±0.102	0.129±0.105	0.077±0.002
Grid	Klein + DiT	0.110±0.047	**0.001±0.001**	0.262±0.041
	Klein + U-Net	**0.105±0.067**	0.002±0.002	0.197±0.037
	Poincaré + DiT	0.330±0.143	0.347±0.102	**0.061±0.101**
MUTAG	Klein + DiT	**0.001±0.001**	**0.001±0.000**	0.131±0.010
	Klein + U-Net	0.002±0.001	**0.001±0.002**	0.144±0.015
	Poincaré + DiT	0.057±0.036	0.111±0.119	**0.130±0.026**
IMDB-B	Klein + DiT	**0.093±0.044**	**0.027±0.019**	**0.074±0.003**
	Klein + U-Net	0.161±0.049	0.144±0.060	0.090±0.012
	Poincaré + DiT	0.122±0.007	0.135±0.043	0.109±0.002

Table 4: Planar/Tree ablations: mean ± sample standard deviation over five seeds. Higher V.U.N. (%) and lower Ratio are better. Bold marks the best displayed mean, including ties. Klein + DiT uses the decoding procedure in Appendix C.6.

Variant	Planar		Tree	
	V.U.N.↑	Ratio↓	V.U.N.↑	Ratio↓
Klein + DiT	**100.000±0.000**	**13.784±3.366**	**100.000±0.000**	**1.160±0.653**
Klein + U-Net	**100.000±0.000**	19.444±3.811	**100.000±0.000**	2.402±0.613
Poincaré + DiT	**100.000±0.000**	16.891±5.739	**100.000±0.000**	1.742±0.373

Table 5: KFM and no-flow controls on the five main datasets: MMD (↓). Entries are mean ± sample standard deviation. KFM uses five independent training seeds; each control uses ten inference seeds from one frozen checkpoint. Bold marks the lowest displayed mean, including ties. Appendix D.4 gives the full protocol.

Dataset	Method	Degree↓	Clustering↓	Spectral↓
Community	KFM	**0.001±0.001**	**0.020±0.010**	**0.061±0.007**
	AE posterior	0.026±0.028	0.090±0.077	0.088±0.020
	VAE prior	0.026±0.026	0.089±0.081	0.088±0.020
Ego	KFM	**0.011±0.005**	**0.019±0.013**	**0.067±0.002**
	AE posterior	0.072±0.028	0.114±0.073	0.073±0.003
	VAE prior	0.070±0.021	0.110±0.073	0.073±0.002
Grid	KFM	0.110±0.047	**0.001±0.001**	**0.262±0.041**
	AE posterior	**0.048±0.028**	0.002±0.002	0.291±0.091
	VAE prior	0.057±0.034	**0.001±0.002**	0.264±0.057
MUTAG	KFM	**0.001±0.001**	**0.001±0.000**	**0.131±0.010**
	AE posterior	0.002±0.003	0.003±0.006	0.141±0.017
	VAE prior	**0.001±0.001**	**0.001±0.002**	0.141±0.017
IMDB-B	KFM	**0.093±0.044**	**0.027±0.019**	**0.074±0.003**
	AE posterior	0.170±0.044	0.110±0.032	0.085±0.007
	VAE prior	0.169±0.051	0.116±0.055	0.087±0.008

Table 6: KFM and no-flow controls on Planar and Tree. All methods use the same structural constraints and candidate-selection procedure. Entries report mean ± sample standard deviation. KFM uses five independent training seeds, as in Table 4; controls use ten inference seeds from one frozen checkpoint. Higher V.U.N. and lower Ratio are better. Bold marks the best displayed mean, including ties.

Method	Planar		Tree	
	V.U.N. (%)↑	Ratio↓	V.U.N. (%)↑	Ratio↓
KFM	**100.00±0.00**	**13.78±3.37**	**100.00±0.00**	**1.16±0.65**
AE posterior	**100.00±0.00**	19.57±3.17	**100.00±0.00**	2.06±0.25
VAE prior	**100.00±0.00**	19.53±2.70	**100.00±0.00**	2.04±0.26
Profile only	**100.00±0.00**	280.34±20.40	**100.00±0.00**	3.42±0.41

conditions and the same generation and evaluation protocol. The posterior control retains instance-specific latent information. On Planar and Tree, a profile-only control replaces learned construction scores with seeded random scores while retaining structural profiles, constraints, and candidate selection. These controls diagnose the conditional pipeline. Full protocols appear in Appendix D.4.

Across the five main datasets (Table 5), the displayed KFM mean is lower in 14 of 15 comparisons with AE posterior, with Grid degree as the exception. Against VAE prior, KFM is lower in 11 comparisons, tied in three, and higher on Grid degree. The ties occur for Grid clustering and MUTAG degree and clustering.

On Planar and Tree, KFM and all controls attain 100% V.U.N. (Table 6). KFM has lower Ratio than all three controls on both datasets. These results support a contribution of learned transport to distribution matching within the conditional pipeline; structural validity is enforced by the construction procedure.

5 Conclusion

Klein Flow Matching learns transport in graph latent space from explicit geodesic velocities, each expressed as a fixed chord direction and a time-dependent scalar factor. Klein-metric regression and Riemannian Heun integration connect this supervision to graph generation. Experiments show strong degree and clustering matching, while spectral fidelity remains a limitation.

AI USE STATEMENT

We used generative AI tools to improve the language and organization of the manuscript. The authors take responsibility for the final content.

REFERENCES

Avishek Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. In *Proceedings of the 12th International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/618c95f4557c15b253fb0e6f548ea0c0-Abstract-Conference.html.

Tianci Bu, Chuanrui Wang, Hao Ma, Haoren Zheng, Xin Lu, and Tailin Wu. GGBall: Graph generative model on Poincaré ball. In *Proceedings of the 14th International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4zRRnDscqn.

Ines Chami. *Representation Learning and Algorithms in Hyperbolic Spaces*. PhD thesis, Stanford University, 2021.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pp. 4869–4880, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0415740eaa4d9decbc8da001d3fd805f-Abstract.html.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *Proceedings of the 12th International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=g7ohDlTITL.

Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4585–4610. PMLR, 2023. URL https://proceedings.mlr.press/v202/chen23k.html.

Antoine Collas, Ce Ju, Nicolas Salvy, and Bertrand Thirion. Riemannian flow matching for brain connectivity matrices via pullback geometry. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/5616112a0120c15bf7d47a6bccc21bc3-Abstract-Conference.html.

Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İlkan Ceylan, Michael Bronstein, and Avishek Joey Bose. Fisher flow matching for generative modeling over discrete data. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/fadec8f2e65f181d777507d1df69b92f-Abstract-Conference.html.

Manfredo P. do Carmo. *Riemannian Geometry*. Birkhäuser, Boston, 1992.

Floor Eijkelboom, Grigory Bartosh, Christian A. Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 11735–11764, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/15b780350b302a1bf9a3bd273f5c15a4-Abstract-Conference.html.

Xingcheng Fu, Yisen Gao, Yuecen Wei, Qingyun Sun, Hao Peng, Jianxin Li, and Xianxian Li. Hyperbolic geometric latent diffusion model for graph generation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14102–14124. PMLR, 2024. URL https://proceedings.mlr.press/v235/fu24c.html.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/dbab2adc8f9d078009ee3fa810bea142-Abstract.html.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL https://www.jmlr.org/papers/v13/gretton12a.html.

Doron Haviv, Aram-Alexandre Pooladian, Dana Pe'er, and Brandon Amos. Wasserstein flow matching: Generative modeling over families of distributions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 22238–22258. PMLR, 2025. URL https://proceedings.mlr.press/v267/haviv25a.html.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. URL https://arxiv.org/abs/2207.12598.

Xiaoyang Hou, Tian Zhu, Milong Ren, Dongbo Bu, Chunming Zhang, Xin Gao, and Shiwei Sun. GGFlow: A graph flow matching method with efficient optimal transport. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=K8RlXtMgzo.

Han Huang, Leilei Sun, Bowen Du, Yanjie Fu, and Weifeng Lv. GraphGDP: Generative diffusion processes for permutation invariant graph generation. In *2022 IEEE International Conference on Data Mining*, pp. 201–210. IEEE, 2022. doi: 10.1109/ICDM54844.2022.00030. URL https://doi.org/10.1109/ICDM54844.2022.00030.

Keyue Jiang, Jiahao Cui, Xiaowen Dong, and Laura Toni. Bures-Wasserstein flow matching for graph generation. In *Proceedings of the 14th International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e3b1291b7529063172d927588d2b03a1-Abstract-Conference.html.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10362–10383. PMLR, 2022. URL https://proceedings.mlr.press/v162/jo22a.html.

Kacper Kapuśniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael Bronstein, Avishek Joey Bose, and Francesco Di Giovanni. Metric flow matching for smooth interpolations on the data manifold. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f381114cf5aba4e45552869863deaaa7-Abstract-Conference.html.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6114.

Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. URL https://arxiv.org/abs/1611.07308. Presented at the NIPS Workshop on Bayesian Deep Learning.

John M. Lee. *Riemannian Manifolds: An Introduction to Curvature*. Springer, New York, 1997.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *Proceedings of the 11th International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Chunyu Luo, Yuankai Luo, Xiao-Ming Wu, and Lei Shi. SimGFM: Simplifying discrete flow matching for graph generation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Z3xgDd5B2L.

Yidan Mao, Jing Gu, Marcus C. Werner, and Dongmian Zou. Klein model for hyperbolic neural networks. In *Proceedings of the 4th (2025) and 3rd (2024) NeurIPS Workshops on Symmetry and Geometry in Neural Representations*, volume 282 of *Proceedings of Machine Learning Research*, pp. 1181–1205. PMLR, 2026. URL https://proceedings.mlr.press/v282/mao26a.html.

Benjamin Kurt Miller, Ricky T. Q. Chen, Anuroop Sriram, and Brandon M. Wood. FlowMM: Generating materials with Riemannian flow matching. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35664–35686. PMLR, 2024. URL https://proceedings.mlr.press/v235/miller24a.html.

Van Khoa Nguyen, Yoann Boget, Frantzeska Lavda, and Alexandros Kalousis. GLAD: Improving latent graph generative modeling with simple quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19695–19702, 2025. doi: 10.1609/aaai.v39i18.34169. URL https://ojs.aaai.org/index.php/AAAI/article/view/34169.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/59dfa2df42d9e3d41f5b02bfc32229dd-Paper.pdf.

Frank Nielsen. A closed-form expression of geodesics in the Klein model of hyperbolic geometry. Online technical note, April 2021. URL https://franknielsen.github.io/KleinGeodesics/ClosedFormKleinGeodesics.pdf.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, October 2023. doi: 10.1109/ICCV51070.2023.00387. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 50269–50326. PMLR, 2025. URL https://proceedings.mlr.press/v267/qin25d.html.

Eric Qu and Dongmian Zou. Autoencoding hyperbolic representation for adversarial generation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=NQi9U0YLW3.

Francesco Ruscelli, Ferdinando Zanchetta, and Rita Fioresi. Cartan flow matching, 2026. URL https://arxiv.org/abs/2605.03588. arXiv preprint.

Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, volume 11139 of *Lecture Notes in Computer Science*, pp. 412–422. Springer, 2018. doi: 10.1007/978-3-030-01418-6_41. URL https://doi.org/10.1007/978-3-030-01418-6_41.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *Proceedings of the 11th International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Lingfeng Wen, Xuan Tang, Mingjie Ouyang, Xiangxiang Shen, Jian Yang, Daxin Zhu, Mingsong Chen, and Xian Wei. Hyperbolic graph diffusion model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15823–15831, 2024. doi: 10.1609/aaai.v38i14.29512. URL https://ojs.aaai.org/index.php/AAAI/article/view/29512.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5708–5717. PMLR, 2018. URL https://proceedings.mlr.press/v80/you18a.html.

Angxiao Yue, Zichong Wang, and Hongteng Xu. ReQFlow: Rectified quaternion flow for efficient and high-quality protein backbone generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73828–73850. PMLR, 2025. URL https://proceedings.mlr.press/v267/yue25c.html.

Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion. In *Advances in Neural Information Processing Systems*, volume 37, pp. 61963–61999, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/718d02a76d69686a36eccc8cde3e6a41-Paper-Conference.pdf.

A Preliminaries of Hyperbolic Geometry

A.1 The Three Analytic Models

The d-dimensional hyperbolic space $\mathbb{H}^d$ is the unique complete, simply connected Riemannian manifold with constant sectional curvature -1 (do Carmo, 1992; Lee, 1997). Its exponential volume growth makes it well suited to representing hierarchical and graph-structured data. Three commonly used analytic models are the hyperboloid model $\mathbb{L}^d$, the Poincaré ball model $\mathbb{B}^d$, and the Klein model $\mathbb{K}^d$. These models are mutually isometric and therefore represent the same intrinsic geometry, although their coordinate expressions differ.

Hyperboloid Model. The hyperboloid model is embedded in Minkowski space $\mathbb{R}^{d,1}$. For $x, y \in \mathbb{R}^{d+1}$, the Lorentzian inner product is

$$\langle x, y \rangle_L = -x_0 y_0 + \sum_{i=1}^{d} x_i y_i.$$

The hyperboloid model is

$$\mathbb{L}^d = \left\{ x \in \mathbb{R}^{d,1} : \langle x, x \rangle_L = -1,\ x_0 > 0 \right\}.$$

Its tangent space at $x \in \mathbb{L}^d$ is

$$T_x \mathbb{L}^d = \left\{ v \in \mathbb{R}^{d,1} : \langle x, v \rangle_L = 0 \right\}.$$

The restriction of the Lorentzian inner product to $T_x \mathbb{L}^d$ is positive definite and defines the Riemannian metric of $\mathbb{L}^d$.

Klein Model. The Klein model is the open unit ball

$$\mathbb{K}^d = \left\{ x \in \mathbb{R}^d : \|x\|_2 < 1 \right\}.$$

Its Riemannian metric tensor is

$$g_x^K = \frac{I}{1 - \|x\|_2^2} + \frac{x x^\top}{(1 - \|x\|_2^2)^2}.$$

Consequently, the induced norm of $v \in T_x \mathbb{K}^d$ is

$$\|v\|_{K,x}^2 = \frac{\|v\|_2^2}{1 - \|x\|_2^2} + \frac{(x^\top v)^2}{(1 - \|x\|_2^2)^2}.$$

The central projection

$$\pi_K : \mathbb{L}^d \to \mathbb{K}^d, \qquad \pi_K(x_0, x) = \frac{x}{x_0},$$

is an isometry. A defining coordinate property of the Klein model is that its geodesics appear as Euclidean line segments.

Poincaré Ball Model. The Poincaré ball has the same coordinate domain as the Klein model,

$$\mathbb{B}^d = \left\{ x \in \mathbb{R}^d : \|x\|_2 < 1 \right\},$$

but is equipped with the conformal metric

$$g_x^B = \frac{4I}{(1 - \|x\|_2^2)^2}.$$

It is isometric to $\mathbb{L}^d$ through stereographic projection:

$$\pi_P : \mathbb{L}^d \to \mathbb{B}^d, \qquad \pi_P(x_0, x) = \frac{x}{1 + x_0}.$$

Unlike the Klein model, the Poincaré model is conformal and therefore preserves Euclidean angles. Its geodesics appear as circular arcs orthogonal to the boundary, with Euclidean diameters as a limiting case.

Figure 3 shows a geodesic on the Lorentz hyperboloid and the projection planes underlying the two ball models (Ganea et al., 2018; Mao et al., 2026). The markers represent equal intrinsic-time samples on the geodesic. Their Poincaré and Klein coordinate realizations are compared in Figure 1: a circular arc orthogonal to the boundary and a Euclidean chord, respectively.

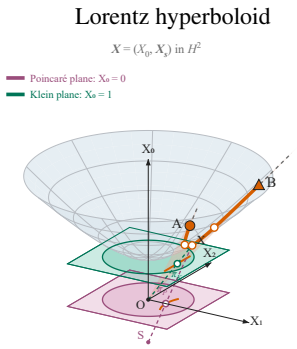

Figure 3: A geodesic on the Lorentz hyperboloid in curvature--1 hyperbolic space. The Poincaré disk lies in the plane $X_0 = 0$ and is obtained by stereographic projection from the south pole $S = (-1, \mathbf{0})$, whereas the Klein disk lies in the plane $X_0 = 1$ and is obtained by central projection from the Lorentz origin $O = (0, \mathbf{0})$. Thus, for a Lorentz point $\mathbf{X} = (X_0, \mathbf{X}_s)$, the corresponding coordinates are $\pi_P(\mathbf{X}) = \mathbf{X}_s/(X_0 + 1)$ and $\pi_K(\mathbf{X}) = \mathbf{X}_s/X_0$. The markers indicate corresponding samples at equal intrinsic-time increments.

A.2 Geodesics and Their Velocities

Let $(\mathcal{M}, g)$ be a Riemannian manifold. A smooth curve $\gamma : [a, b] \to \mathcal{M}$ is a geodesic if

$$\nabla_{\dot\gamma}\dot\gamma = 0.$$

In local coordinates, this equation becomes

$$\ddot\gamma^k(t) + \sum_{i,j} \Gamma^k_{ij}(\gamma(t))\dot\gamma^i(t)\dot\gamma^j(t) = 0,$$

where

$$\Gamma^k_{ij} = \frac{1}{2}\sum_{\ell=1}^{d} g^{k\ell}\left(\frac{\partial g_{j\ell}}{\partial x^i} + \frac{\partial g_{i\ell}}{\partial x^j} - \frac{\partial g_{ij}}{\partial x^\ell}\right).$$

The exponential map at $x \in \mathcal{M}$ is defined by

$$\exp_x(v) = \gamma_{x,v}(1),$$

where $\gamma_{x,v}$ is the geodesic satisfying $\gamma_{x,v}(0) = x$ and $\dot\gamma_{x,v}(0) = v$. The logarithmic map is its local inverse:

$$\log_{x_0}(x_1) = v \quad \Longleftrightarrow \quad \exp_{x_0}(v) = x_1.$$

Whenever the logarithmic map is uniquely defined, the geodesic between two endpoints is

$$\gamma_{x_0,x_1}(t) = \exp_{x_0}\left(t \log_{x_0}(x_1)\right), \qquad t \in [0, 1].$$

On a general manifold this expression is local or requires the choice of a minimizing logarithm branch. Hyperbolic space is complete, simply connected, and has negative curvature, so the minimizing geodesic between any two points is globally unique.

We use

$$\dot x_t := \frac{d}{dt}x_t$$

for the coordinate representation of the tangent velocity along a path x_t.

Hyperboloid Model. Let $x \in \mathbb{L}^d$ and $v \in T_x\mathbb{L}^d$ satisfy $\|v\|_{L,x} = 1$. The corresponding unit-speed geodesic is

$$\gamma^L_{x,v}(t) = \cosh(t)x + \sinh(t)v.$$

We use the standard distance, exponential map, and logarithmic map of the Lorentz model (Chami, 2021):

$$
\begin{aligned}
d_L(x,y) &= \operatorname{arcosh}\left(-\langle x,y\rangle_L\right),\\
\exp_x^L(v) &= \cosh(\|v\|_{L,x})x + \sinh(\|v\|_{L,x})\frac{v}{\|v\|_{L,x}},\\
\log_x^L(y) &= d_L(x,y)\frac{y+\langle x,y\rangle_L x}{\|y+\langle x,y\rangle_L x\|_{L,x}}.
\end{aligned}
$$

The expressions involving division by a tangent norm are understood by continuous extension at the zero vector.

Let $\rho = d_L(x_0,x_1)$. The constant-speed geodesic and its velocity are

$$
\begin{aligned}
x_t &= \frac{\sinh((1-t)\rho)}{\sinh\rho}x_0 + \frac{\sinh(t\rho)}{\sinh\rho}x_1,\\
\dot{x}_t &= \frac{\rho}{\sinh\rho}\left(\cosh(t\rho)x_1 - \cosh((1-t)\rho)x_0\right).
\end{aligned}
$$

For $\rho = 0$, these formulas are understood as the constant path $x_t = x_0$ with $\dot{x}_t = \mathbf{0}$.

Klein Model. A distinguishing property of the Klein model is that its unparameterized geodesics are represented by Euclidean line segments. Using the isometry between the Lorentz hyperboloid and the Klein ball, Mao et al. (2026) derive the following unit-speed geodesic. For $x \in \mathbb{K}^d$ and $v \in T_x\mathbb{K}^d$ satisfying $\|v\|_{K,x} = 1$,

$$
\gamma_{x,v}^K(t) = x + \frac{\sinh(t)v}{\cosh(t) + \lambda_x^2(x^\top v)\sinh(t)}, \qquad \lambda_x = \frac{1}{\sqrt{1-\|x\|_2^2}}. \tag{10}
$$

For an arbitrary tangent vector $v \in T_x\mathbb{K}^d$, let

$$
r = \|v\|_{K,x}.
$$

Reparameterizing the unit-speed geodesic in Eq. (10) by r yields the exponential map, while solving the corresponding endpoint relation yields the logarithmic map (Mao et al., 2026):

$$
\begin{aligned}
\exp_x^K(v) &= x + \frac{\frac{\sinh(r)}{r}v}{\cosh(r) + \lambda_x^2(x^\top v)\frac{\sinh(r)}{r}},\\
\log_x^K(y) &= d_K(x,y)\frac{y-x}{\|y-x\|_{K,x}}.
\end{aligned}
$$

The exponential map at $v = \mathbf{0}$ and the logarithmic map at $y = x$ are defined by continuous extension.

Under the same Lorentz–Klein isometry, the Klein geodesic distance is (Mao et al., 2026)

$$
d_K(x,y) = \operatorname{arcosh}\left(\frac{1-x^\top y}{\sqrt{1-\|x\|_2^2}\sqrt{1-\|y\|_2^2}}\right).
$$

Substituting the endpoint logarithmic map into the exponential map gives the conditional path used by KFM. For distinct $x_0, x_1 \in \mathbb{K}^d$, define

$$
\begin{aligned}
\Delta &= x_1 - x_0, & r_i &= \sqrt{1-\|x_i\|_2^2},\\
S^2 &= (1-\|x_0\|_2^2)\|\Delta\|_2^2 + (\Delta^\top x_0)^2, & \rho &= d_K(x_0,x_1),\\
u &= \frac{(1-\|x_0\|_2^2)\Delta}{S}, & \alpha &= \frac{x_0^\top u}{1-\|x_0\|_2^2}.
\end{aligned} \tag{11}
$$

The distance can equivalently be written as

$$
\rho = \operatorname{arcosh}\left(\frac{1-x_0^\top x_1}{r_0 r_1}\right).
$$

Substituting $t \log_{x_0}^K(x_1)$ into the exponential map gives

$$x_t = x_0 + \frac{u}{\coth(t\rho) + \alpha}. \tag{12}$$

Differentiation with respect to time gives

$$\dot{x}_t = \frac{\rho u \left[\coth^2(t\rho) - 1\right]}{\left[\coth(t\rho) + \alpha\right]^2}. \tag{13}$$

Both formulas are understood by continuous extension when $\rho = 0$ or $t = 0$. In particular, $\lim_{t\to 0} x_t = x_0$ and $x_1 = \exp_{x_0}^K(\log_{x_0}^K(x_1))$.

Equivalent Geometric Form of the Velocity. Because the Klein geodesic in Eq. (12) traces a Euclidean chord, its coordinate velocity is parallel to $x_1 - x_0$. Moreover, the exponential-map parameterization has constant Riemannian speed $\rho = d_K(x_0, x_1)$. Therefore,

$$\dot{x}_t = \rho \frac{x_1 - x_0}{\|x_1 - x_0\|_{K,x_t}}.$$

Expanding the tangent norm using the Klein metric gives

$$\begin{aligned} \dot{x}_t &= \rho \frac{(1 - \|x_t\|_2^2)(x_1 - x_0)}{\sqrt{(1 - \|x_t\|_2^2)\|x_1 - x_0\|_2^2 + \left[(x_1 - x_0)^\top x_t\right]^2}} \\ &= \rho \frac{(1 - \|x_t\|_2^2)(x_1 - x_0)}{\sqrt{(1 - x_0^\top x_1)^2 - (1 - \|x_0\|_2^2)(1 - \|x_1\|_2^2)}}. \end{aligned} \tag{14}$$

The denominator in the final expression is equal to S defined in Eq. (11). Hence, Eq. (14) is algebraically equivalent to the differentiated expression in Eq. (13). Taking its Klein norm gives

$$\|\dot{x}_t\|_{K,x_t} = d_K(x_0, x_1),$$

for every $t \in [0, 1]$, with endpoint values understood by continuous extension. Thus, the Euclidean coordinate direction remains fixed, whereas its scalar magnitude varies along the path.

Equivalence to the chord formula in the main text. Using the notation of Eq. (3), $u = a\Delta/S$ and $\alpha = b/S$. Multiplying numerator and denominator of Eq. (12) by $S\sinh(t\rho)$ gives

$$x_t = x_0 + \frac{a\sinh(t\rho)}{B_t}\Delta, \qquad B_t = S\cosh(t\rho) + b\sinh(t\rho). \tag{15}$$

This is the simplified constant-speed chord parameterization of Nielsen (2021). Differentiating and using $\cosh^2(t\rho) - \sinh^2(t\rho) = 1$ yields

$$\dot{x}_t = \frac{aS\rho}{B_t^2}\Delta = \rho\frac{1 - \|x_t\|_2^2}{S}\Delta, \tag{16}$$

where $1 - \|x_t\|_2^2 = aS^2/B_t^2$. These expressions agree with Eq. (4) and Eq. (14). For distinct endpoints $S > |b|$, so $B_t > 0$ on $[0, 1]$. At $t = 0$, the velocity is $\rho a\Delta/S = \log_{x_0}^K(x_1)$; coincident endpoints define a constant path with zero velocity.

Poincaré Ball Model. For comparison, we use the standard Möbius operations and exponential and logarithmic maps of the Poincaré ball (Ganea et al., 2018). The Möbius addition on $\mathbb{B}^d$ is

$$x \oplus_M y = \frac{(1 + 2x^\top y + \|y\|_2^2)x + (1 - \|x\|_2^2)y}{1 + 2x^\top y + \|x\|_2^2\|y\|_2^2}.$$

The exponential and logarithmic maps are

$$\begin{aligned} \exp_x^B(v) &= x \oplus_M \left[\tanh\left(\frac{\|v\|_2}{1 - \|x\|_2^2}\right)\frac{v}{\|v\|_2}\right], \\ \log_x^B(y) &= (1 - \|x\|_2^2)\,\mathrm{artanh}\left(\|(-x) \oplus_M y\|_2\right)\frac{(-x) \oplus_M y}{\|(-x) \oplus_M y\|_2}. \end{aligned}$$

The Poincaré distance is

$$d_B(x,y) = \operatorname{arcosh}\left(1 + \frac{2\|x-y\|_2^2}{(1-\|x\|_2^2)(1-\|y\|_2^2)}\right).$$

Given $x_0, x_1 \in \mathbb{B}^d$, define

$$z = (-x_0) \oplus_M x_1, \qquad r = \|z\|_2, \qquad e = \frac{z}{r}, \qquad a = \operatorname{artanh}(r).$$

For $r > 0$, the constant-speed Poincaré geodesic is

$$x_t = x_0 \oplus_M [b_t e], \qquad b_t = \tanh(ta).$$

It satisfies

$$\|\dot{x}_t\|_{B,x_t} = d_B(x_0, x_1) = 2a.$$

For an explicit coordinate velocity, define

$$A_t = 1 + 2b_t x_0^\top e + b_t^2, \qquad B = 1 - \|x_0\|_2^2,$$
$$C_t = 1 + 2b_t x_0^\top e + b_t^2 \|x_0\|_2^2, \qquad \mathbf{N}_t = A_t x_0 + B b_t e.$$

Then $x_t = \mathbf{N}_t / C_t$, with

$$\dot{b}_t = a(1 - b_t^2), \qquad \dot{A}_t = 2\dot{b}_t(x_0^\top e + b_t),$$
$$\dot{C}_t = 2\dot{b}_t\left(x_0^\top e + b_t\|x_0\|_2^2\right), \qquad \dot{\mathbf{N}}_t = \dot{A}_t x_0 + B\dot{b}_t e.$$

The coordinate velocity is therefore

$$\dot{x}_t = \frac{C_t\dot{\mathbf{N}}_t - \mathbf{N}_t\dot{C}_t}{C_t^2}.$$

When $r = 0$, the path and velocity are defined by continuity as $x_t = x_0$ and $\dot{x}_t = \mathbf{0}$.

Comparison of hyperbolic models. The Lorentz, Poincaré, and Klein models represent the same intrinsic geometry. Each admits closed-form geodesic paths and velocities. The Poincaré expression above can be evaluated analytically; it does not require automatic differentiation of Möbius addition at training time. The specific convenience of Klein coordinates is the representation $\dot{x}_t = \eta_t(x_1 - x_0)$ with a fixed chord direction. This property concerns individual endpoint-conditioned paths, not the trajectories of the learned marginal field. Section 4.3 evaluates the resulting implementation together with its learned representation.

B Riemannian Conditional Flow Matching

Appendix A establishes the geometric operations and closed-form geodesics used by KFM. This appendix presents the probabilistic framework that connects those paths and their velocities to a simulation-free training objective, and then describes Riemannian Heun integration for generation. The construction follows Euclidean conditional flow matching (Lipman et al., 2023) and its extension to Riemannian manifolds (Chen & Lipman, 2024). The density-based derivation below describes the ideal construction with an unprojected base distribution and exact geodesic paths. The radius projection used in the implementation can introduce mass on a sphere, so the projected distributions need not have densities with respect to Riemannian volume. The regression decomposition remains valid for their probability laws when the stated moment conditions hold; a density-based transport argument does not by itself establish exact sampling for the projected implementation.

B.1 Principles of Riemannian Conditional Flow Matching

Flows and Probability Paths on Manifolds. Let $(\mathcal{M}, g)$ be a complete, connected, smooth Riemannian manifold, and let $d\mathrm{vol}_g$ denote its Riemannian volume measure. A time-dependent vector field $v_t(x) \in T_x\mathcal{M}$ with local Lipschitz regularity and solutions existing on $[0,1]$ defines a flow $\Phi_t : \mathcal{M} \to \mathcal{M}$ through

$$\frac{d}{dt}\Phi_t(x) = v_t\left(\Phi_t(x)\right), \qquad \Phi_0(x) = x.$$

If $x_0 \sim p_0$ and $x_t = \Phi_t(x_0) \sim p_t$, and these laws admit densities with respect to $d\mathrm{vol}_g$, conservation of probability mass is described by

$$\partial_t p_t(x) + \mathrm{div}_g\left(p_t v_t\right)(x) = 0.$$

In local coordinates,

$$\mathrm{div}_g\, v = \frac{1}{\sqrt{|g(x)|}} \sum_{i=1}^{d} \frac{\partial}{\partial x_i}\left(\sqrt{|g(x)|}\, v^i\right).$$

Geodesic Conditional Paths. For this construction, assume that $\mathcal{M}$ is a Hadamard manifold, as is hyperbolic space, so the logarithmic map is globally unique. Let $x_0 \sim p_0$ and $x_1 \sim q$ be independent base and target samples. For a differentiable increasing scheduler $\kappa : [0,1] \to [0,1]$ satisfying $\kappa(0) = 0$ and $\kappa(1) = 1$, define

$$x_t = \exp_{x_0}\left(\kappa(t) \log_{x_0}(x_1)\right).$$

Its tangent velocity is

$$\dot{x}_t = \dot{\kappa}(t) P_{x_0 \to x_t} \log_{x_0}(x_1),$$

where $P_{x_0 \to x_t}$ denotes parallel transport along the geodesic. For $\kappa(t) = t$,

$$\|\dot{x}_t\|_{g,x_t} = d_g(x_0, x_1).$$

Marginalization of conditional vector fields. Let q be a target probability measure, including an empirical measure, and let $p_t(x \mid x_1)$ be the density of the conditional path for $0 < t < 1$. Define

$$p_t(x) = \int_{\mathcal{M}} p_t(x \mid x_1)\, q(dx_1), \qquad u_t(x \mid x_1) = \mathbb{E}[\dot{x}_t \mid x_t = x, x_1].$$

Where $p_t(x) > 0$, the marginal velocity is

$$\bar{u}_t(x) = \frac{\int_{\mathcal{M}} u_t(x \mid x_1) p_t(x \mid x_1)\, q(dx_1)}{p_t(x)} = \mathbb{E}[\dot{x}_t \mid x_t = x].$$

Assuming integrability and sufficient regularity to interchange the integral with the weak continuity equation, the conditional equations give $\partial_t p_t + \mathrm{div}_g(p_t \bar{u}_t) = 0$. For $q = M^{-1} \sum_i \delta_{x_{1,i}}$, the integrals over q are finite sums; no density of the endpoint measure is assumed. Conditioning on a structural code c gives the same identities with $q(dx_1 \mid c)$ and $p_t(x \mid c)$. The empirical endpoint law is a limiting target at $t = 1$; these identities do not assert exact collapse to atoms under a smooth finite-time learned flow.

Riemannian Conditional Flow Matching. The ideal Riemannian flow-matching objective is

$$\mathcal{L}_{\mathrm{RFM}}(\theta) = \mathbb{E}\left[\|v_\theta(x_t, t) - \bar{u}_t(x_t)\|^2_{g,x_t}\right].$$

Because the marginal velocity is intractable, Riemannian conditional flow matching instead uses

$$\mathcal{L}_{\mathrm{RCFM}}(\theta) = \mathbb{E}\left[\|v_\theta(x_t, t) - \dot{x}_t\|^2_{g,x_t}\right].$$

For finite second moments in the Riemannian norm, the conditional-expectation identity gives

$$\mathcal{L}_{\mathrm{RCFM}}(\theta) = \mathcal{L}_{\mathrm{RFM}}(\theta) + \Delta_{\mathrm{FM}}, \qquad \nabla_\theta \mathcal{L}_{\mathrm{RCFM}} = \nabla_\theta \mathcal{L}_{\mathrm{RFM}},$$

where

$$\Delta_{\mathrm{FM}} = \mathbb{E}\left[\|\dot{x}_t - \bar{u}_t(x_t)\|^2_{g,x_t}\right]$$

does not depend on θ. The gradient identity additionally assumes that differentiation and expectation can be interchanged.

Wrapped Gaussian Base Distribution. The ideal KFM base distribution wraps an isotropic Gaussian in the tangent space at the Klein origin:

$$\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 I), \qquad x_0 = \exp_{\mathbf{0}}^K(\epsilon).$$

This is the push-forward of a Euclidean Gaussian under the origin exponential map. KFM requires samples from this distribution but does not require evaluation of its density.

Latent Standardization. Let $z^\star$ denote the deterministic Klein latent produced by the frozen encoder. Define

$$y = \log_{\mathbf{0}}^K(z^\star), \qquad \widetilde{y} = (y - m) \oslash s, \qquad x_1 = \exp_{\mathbf{0}}^K(\widetilde{y}).$$

Because the origin exponential and logarithmic maps are global inverses on hyperbolic space, this transformation produces a valid Klein point whenever every component of s is positive. It is an invertible preprocessing transformation in exact arithmetic, but not a hyperbolic isometry. The implementation additionally clips points to a smaller ball; that clipping is not invertible.

After generation, standardization is inverted by

$$z_{\text{gen}} = \exp_{\mathbf{0}}^K\left(\log_{\mathbf{0}}^K(x_{\text{gen}}) \odot s + m\right).$$

Coordinate Representation of Tangent Vectors. The Klein ball is an open subset of $\mathbb{R}^d$, so every tangent space $T_x\mathbb{K}^d$ is canonically identified with $\mathbb{R}^d$. The velocity network can therefore output an ordinary d-dimensional coordinate vector. This does not make the geometry Euclidean: velocity errors are measured with the position-dependent Klein metric, and manifold updates are performed with the Klein exponential map.

KFM Objective and Structural Conditioning. Let (x_1, c) follow the empirical joint distribution of standardized graph latents and structural conditions, and sample x_0 independently from the wrapped Gaussian base. With the linear scheduler $\kappa(t) = t$, KFM optimizes

$$\mathcal{L}_{\text{KFM}}(\theta) = \mathbb{E}\left[\|v_\theta(x_t, t, c) - \dot{x}_t\|_{K,x_t}^2\right].$$

The closed-form Klein path and velocity make the construction of training pairs simulation-free. In practice, $t \sim \mathcal{U}(\varepsilon_t, 1 - \varepsilon_t)$, with $\varepsilon_t = 10^{-4}$, to avoid endpoint instabilities in expressions involving $\coth(t\rho)$ and $\operatorname{csch}(t\rho)$.

Classifier-Free Guidance. During training, c is replaced by the zero vector with probability p_{drop}. At inference time, the guided vector field is

$$v_w(x, t, c) = v_\theta(x, t, \mathbf{0}) + w\left[v_\theta(x, t, c) - v_\theta(x, t, \mathbf{0})\right].$$

Both predictions belong to the same tangent space $T_x\mathbb{K}^d$, so their linear combination is a valid tangent vector. Classifier-free guidance modifies the learned vector field at inference time. The choice $w = 1$ recovers the conditional field; the probability-path identity does not generally hold for $w \neq 1$.

B.2 Riemannian Heun Integration

Generation requires solving

$$\dot{x}_t = v_w(x_t, t, c), \qquad x_0 \sim p_0.$$

For step size h and $t_{k+1} = t_k + h$, compute the predictor:

$$v_k = v_w(x_k, t_k, c),$$
$$\widetilde{x}_{k+1} = \exp_{x_k}^K(h v_k).$$

Evaluate the endpoint velocity:

$$\widetilde{v}_{k+1} = v_w\left(\widetilde{x}_{k+1}, t_{k+1}, c\right).$$

Transport it back to $T_{x_k}\mathbb{K}^d$ before averaging:

$$v_{k+1}^{\leftarrow} = P_{\widetilde{x}_{k+1} \to x_k}^K\left(\widetilde{v}_{k+1}\right).$$

The corrected update is

$$x_{k+1} = \exp_{x_k}^K\left[\frac{h}{2}\left(v_k + v_{k+1}^{\leftarrow}\right)\right]. \tag{17}$$

Parallel Transport Through the Lorentz Isometry. We compute exact Klein parallel transport through the Lorentz isometry

$$\iota(x) = (\lambda_x, \lambda_x x)\,.$$

Its differential is

$$D\iota_x[v] = \left(\lambda_x^3 x^\top v,\ \lambda_x v + \lambda_x^3 (x^\top v) x\right).$$

Let

$$\mathbf{X} = \iota(x), \qquad \mathbf{Y} = \iota(y), \qquad \mathbf{V} = D\iota_x[v].$$

Lorentz parallel transport along the unique geodesic is

$$\mathbf{W} = P^L_{\mathbf{X}\to\mathbf{Y}}(\mathbf{V}) = \mathbf{V} + \frac{\langle \mathbf{Y}, \mathbf{V}\rangle_L}{1 - \langle \mathbf{X}, \mathbf{Y}\rangle_L}(\mathbf{X} + \mathbf{Y}).$$

Writing $\mathbf{W} = (W_0, \mathbf{W}_s)$, the transported Klein tangent vector is

$$P^K_{x\to y}(v) = \frac{\mathbf{W}_s - W_0 y}{\lambda_y}.$$

Because the construction is induced by an isometry, it preserves the Riemannian inner product:

$$\left\langle P^K_{x\to y}(u), P^K_{x\to y}(v)\right\rangle_{K,y} = \langle u, v\rangle_{K,x}.$$

Order of Accuracy. Assume a twice continuously differentiable field on a compact interior region containing the exact and numerical trajectories, bounded derivatives and Lipschitz constants sufficient for stability, exact geometric operations, and inactive projection. A covariant Taylor expansion gives

$$P_{x(t+h)\to x(t)} v(x(t+h), t+h) = v + h\,(\partial_t v + \nabla_v v) + O(h^2).$$

The predictor has endpoint error $O(h^2)$, which perturbs the transported velocity by $O(h^2)$. Thus the averaged increment in Eq. (17) has local error $O(h^3)$, yielding global error $O(h^2)$ under these assumptions.

Numerical Stabilization. The exact Klein exponential map stays inside the unit ball. In finite precision, we enforce radius $1 - \varepsilon_{\text{ball}}$ by

$$\text{proj}_{\mathbb{K}}(x) = \begin{cases} x, & \|x\|_2 \le 1 - \varepsilon_{\text{ball}}, \\ (1 - \varepsilon_{\text{ball}})\dfrac{x}{\|x\|_2}, & \text{otherwise.} \end{cases}$$

The boundary margin is $\varepsilon_{\text{ball}} = 4 \times 10^{-3}$ in float32 and 10^{-5} in float64. Selected denominators and latent standard deviations are floored at 10^{-15} and 10^{-4}, respectively. The float32 radius cap 0.996 corresponds to origin-tangent norm $\operatorname{artanh}(0.996) \approx 3.1063$. Projection changes the base and target distributions and discards radial information; Appendix D.1 measures its frequency and reconstruction effects. The second-order guarantee excludes these projections. Interior-time sampling reduces endpoint instabilities in the coth formula but does not resolve cancellation or coincident endpoints.

C Graph Representation, Decoding, and Evaluation

The standard pipeline serves Community, Ego, MUTAG, IMDB-B, Community-small, and Ego-small. Grid, Planar, and Tree use a structural pipeline with a GIN encoder, structural conditioning, and degree-aware decoding. Both supply deterministic targets and conditions to KFM.

C.1 Input Features and Node Ordering

Graphs are padded to $N_{\max}$ slots with a binary node mask a_i. For a graph with n nodes, the structural input for node i contains its degree divided by $\max(n-1, 1)$, log-degree $\log(1 + \deg(i))$, clustering coefficient, core number divided by the maximum core number (floored at one), PageRank, a valid-node indicator, and normalized-Laplacian eigenvector features. The latter are computed from the smallest eigenvalues, omitting the first returned eigenvector when multiple vectors are available;

unused positional channels are zero-padded. Although the feature constructor accepts node attributes, the MUTAG and IMDB-B loaders supply only adjacency matrices. We generate topology, excluding atom/bond types and other attributes. Message-passing self-loops are excluded from reconstruction targets.

Planar/Tree use structural BFS, ranking roots and neighbors lexicographically by decreasing degree, core number, and closeness centrality, then increasing node identifier. Traversal restarts at the highest-ranked unvisited node. Ordinary BFS starts each component at its smallest node identifier; datasets without BFS retain the supplied order. Reconstruction compares adjacency slots without graph matching. These orderings are not isomorphism-invariant, and spectral features depend on the numerical eigenvector signs and bases.

C.2 Encoder and Structural Conditions

Let $H = \max(128, \min(256, 4d))$ be the encoder hidden width. A node-feature projection uses a linear layer of width 64, GELU, layer normalization, and a linear map to H. The standard pipeline applies residual GraphConv blocks with symmetric degree normalization. The structural pipeline uses sum-aggregation GIN blocks with a learned self-feature coefficient and a two-layer $H \to 2H \to H$ MLP. Each block includes a GELU residual update with layer normalization, masked mean pooling followed by an MLP that broadcasts graph context back to valid nodes, and a residual feed-forward network with hidden width $4H$. Padding is masked after each block.

The graph readout concatenates attention-weighted, masked mean, and masked maximum pooling. Attention scores come from an $H \to H \to 1$ MLP and are normalized over valid nodes. A $3H \to 2H \to H$ projection yields a graph representation, from which separate heads predict the tangent mean μ_G, log standard deviation $\ell_G = \log \sigma_G$, a learned condition c_G^{learn}, and six auxiliary graph statistics. The graph convolutions and pooling operate on Euclidean features; the latent is placed in the Klein ball by the origin exponential map.

The structural pipeline appends an embedding of a 40-dimensional profile: eight scalars (node/edge counts, density, mean degree, second through fourth degree moments, and maximum degree) and a normalized 32-bin degree histogram on $[0, \max(n-1, 1)]$. Scalars are standardized using training statistics, replacing standard deviations below 10^{-6} by one. Layer normalization and a $40 \to 2d_s \to d_s$ MLP with GELU produce c_G^{struct}. Thus $c_G = [c_G^{\text{learn}} \,\|\, c_G^{\text{struct}}]$ and $d_c = d_{\text{learn}} + d_s$; the standard pipeline uses only c_G^{learn}. Conditions are learned graph codes, not class labels.

Autoencoder training maps samples $u = \mu_G + \exp(\ell_G) \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, to the Klein ball. Flow targets instead use evaluation-mode posterior means, $z_G^\star = \exp_0^K(\mu_G)$, with the projection in Appendix B.2.

C.3 Masked Slot Decoder

The decoder input is $r = [\log_0^K(z) \,\|\, c]$. Each of the $N_{\max}$ slots has a learned embedding $e_i \in \mathbb{R}^{H_D}$. Two MLPs map r to a global context $b(r)$ and FiLM parameters $(\delta(r), \gamma(r))$. The initial slot representation is

$$h_i = (1 + \gamma(r)) \odot e_i + \delta(r) + b(r),$$

followed by a residual layer-normalized $H_D \to 2H_D \to H_D$ MLP. A node head maps each slot through widths $H_D/2$ and one to a presence logit l_i^V. For edges, four linear maps form

$$e_{ij} = U_s h_i + U_s h_j + (U_p h_i) \odot (U_p h_j) + |U_d h_i - U_d h_j| + U_g r.$$

A $16 \to 32 \to 1$ MLP maps e_{ij} to edge logits l_{ij}^E. The symmetric pair construction is followed by logit symmetrization and diagonal exclusion. A graph head predicts the encoder's six auxiliary statistics; the structural decoder adds an MLP with softplus for nonnegative slot degrees. Sigmoids give node and edge probabilities.

C.4 Autoencoder Training Objective

For a minibatch of B graphs, let A_{bij} be the padded binary adjacency and a_{bi} the node mask. Define $Q_{bij} = a_{bi} a_{bj} \mathbf{1}_{i \neq j}$, $P = \max(1, \sum_{bij} Q_{bij} A_{bij})$, and $\omega_+ = \max(1, (\max(1, \sum_{bij} Q_{bij}) - P)/P)$.

The edge and node reconstruction losses are

$$\mathcal{L}_E = \frac{\sum_{bij} Q_{bij}\,\mathrm{BCE}_{\omega_+}(l^E_{bij}, A_{bij})}{\max(1, \sum_{bij} Q_{bij})},$$

$$\mathcal{L}_V = \frac{1}{BN_{\max}} \sum_{bi} \mathrm{BCE}(l^V_{bi}, a_{bi}).$$

BCE uses logits, with positive edge weight ω_+. Edge loss includes both off-diagonal orientations; node loss includes padded slots.

For each graph, the six-dimensional auxiliary target is $s_G = (n, m, \mathrm{density}, \overline{\deg}, \overline{\deg^2}, \overline{\deg^3})$, computed on valid nodes and off-diagonal edges. The statistic loss is the sum of mean smooth-L_1 losses for the encoder and decoder predictions against s_G; these six targets are not standardized in this loss. The Gaussian regularizer is normalized by both batch size and latent dimension:

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{2Bd} \sum_{b=1}^{B} \sum_{j=1}^{d} \left(\mu^2_{bj} + \exp(2\ell_{bj}) - 1 - 2\ell_{bj}\right).$$

The implemented training objective is

$$\mathcal{L}_{\mathrm{AE}} = \mathcal{L}_E + 0.25\mathcal{L}_V + 0.1\mathcal{L}_{\mathrm{stat}} + 0.05\mathcal{L}_{\mathrm{KL}} + \lambda_{\mathrm{deg}}\mathcal{L}_{\mathrm{deg}} + \lambda_{\mathrm{hist}}\mathcal{L}_{\mathrm{hist}}. \tag{18}$$

The final two terms are absent in the standard pipeline. In the structural pipeline, $\mathcal{L}_{\mathrm{deg}}$ is mean smooth-L_1 error between predicted and target degrees over valid nodes. $\mathcal{L}_{\mathrm{hist}}$ is the mean L_1 distance between normalized soft degree histograms of the predicted and target adjacencies. It uses 32 Gaussian bins on a shared degree scale with bandwidth $0.75D_{\max}/31$ (floored at 10^{-3}). The degree-axis upper bound uses the training-profile mean maximum degree plus two standard deviations, floored at one. The dataset configuration supplies λ_{deg} and λ_{hist}.

On constrained benchmarks, the histogram loss calibrates edge probabilities to the target edge count using a detached threshold from 40 bisection steps and a temperature-scaled sigmoid. Gradients pass through the shifted logits; weighted edge BCE remains unchanged. No additional graph-kernel matching loss is used.

C.5 Velocity Network and Numerical Settings

The DiT linearly projects the $2d + 2$ features from Section 3.3 to four tokens of width H_F and adds learned positional embeddings. MLP embeddings of sinusoidal time features and condition c are summed to modulate attention and feed-forward blocks through AdaLN-Zero. Feed-forward width is $4H_F$. After adaptive layer normalization, tokens are flattened and projected to a d-dimensional velocity. The output projection and modulation gates are zero-initialized; Table 7 gives width, depth, and head count.

Flow training samples $t \sim \mathcal{U}(10^{-4}, 1 - 10^{-4})$ and uses zero-code classifier-free dropout. Tangent normalization uses only training latents, with sample standard deviations floored at 10^{-4}. Both stages use the optimizer settings in Section 4. Validation metrics select dataset configurations; checkpoint selection within each run follows separate rules. The standard pipeline selects the encoder by lowest observed training-minibatch loss and uses the final flow epoch. The structural pipeline selects checkpoints by validation metrics, with the Planar/Tree criterion in Appendix C.7.

C.6 Graph Decoding and Candidate Selection

Standard pipeline. The node count is the number of presence probabilities above 0.5, falling back to the rounded node-count prediction clamped to $[1, N_{\max}]$ if zero. Retain that many highest-probability slots and threshold off-diagonal edges at 0.5. Community-small and Ego-small retain the full graph. Other standard datasets remove isolated nodes, keep the largest connected component, and discard empty outputs.

Structural conditions and graph sizes. Sampled training conditions supply node/edge counts and degree histograms; these counts replace decoder predictions. When enabled, isotropic Gaussian noise perturbs the entire condition, including its structural embedding. Planar/Tree condition and profile banks are aligned by training-graph index.

Grid. Retain the highest-probability node slots and select high-scoring edges using predicted degrees and an edge budget. Edges added between components repair connectivity within configured budget tolerances. The Grid recipe generates four times the requested number of candidates for selection against a profile fitted on training graphs.

Tree and Planar. Blend predicted degrees with the sampled training degree histogram and round to positive integer targets summing to twice the edge budget. Tree uses a score-weighted Prüfer construction to realize the degree sequence with $n-1$ edges. Planar builds a scored spanning tree, then adds edges subject to planarity up to the sampled budget. Its target degrees guide selection without an exact realization guarantee. Optional noise perturbs construction scores.

For Planar and Tree, eight times the requested number of graphs are generated. Candidates are ranked by their distance from the training profile: absolute standardized deviations in node count, edge count, density, mean degree, and maximum degree have weights $1, 1, 1, 0.5, 0.5$, respectively, and the L_1 degree-histogram distance has weight 0.25. Selection favors non-isomorphic candidates, then fills remaining slots from the ranked pool without using the novelty reference set. Decoding guarantees validity; uniqueness and novelty are evaluated after selection.

C.7 Evaluation Details

MMD estimator. The implementation reports the biased squared MMD estimator, retaining the conventional label "MMD" in the tables:

$$\widehat{\mathrm{MMD}}^2 = \frac{1}{R^2}\sum_{i,i'=1}^{R} k(f_i, f_{i'}) + \frac{1}{S^2}\sum_{j,j'=1}^{S} k(\hat{f}_j, \hat{f}_{j'}) - \frac{2}{RS}\sum_{i=1}^{R}\sum_{j=1}^{S} k(f_i, \hat{f}_j).$$

Here f is a per-graph statistic vector. Degree uses integer-count bins, clustering uses 100 bins on $[0, 1]$, and spectral statistics use 200 bins on $[-10^{-5}, 2]$ for normalized-Laplacian eigenvalues. Histograms are normalized by their sums plus 10^{-6} in the MMD helper. For the five main datasets, the kernel is $k(f, g) = \exp[-D(f, g)^2/(2\sigma^2)]$, where D is earth mover's distance. Degree and spectral kernels use unit bin-index distances and $\sigma = 1$; clustering divides the bin-index distance by 100 and uses $\sigma = 0.1$. Before evaluation, each reference graph is paired with the generated graph closest in node count, allowing reuse.

For Community-small and Ego-small, the degree and clustering kernels instead use $D(f, g) = \|f - g\|_1/2$, with $\sigma = 1$ and 0.1, respectively. Orbit features are the 15 ORCA node-orbit counts for graphlets with at most four nodes, summed across nodes and divided by node count. Their kernel uses the same half-L_1 distance with $\sigma = 30$, without histogram normalization. These datasets do not apply the main-dataset size-matching step.

V.U.N. and Ratio. For S generated graphs, V.U.N. is the number of distinct generated isomorphism classes that are valid and have no isomorphic representative in the training set, divided by S. We report this joint fraction as a percentage. Isomorphism tests use unlabelled simple graphs. Valid outputs are nonempty connected planar graphs for Planar and nonempty trees for Tree.

Ratio averages normalized squared-MMD distances for degree, clustering, orbit, spectral, and wavelet statistics. These use half-L_1 Gaussian kernels. Wavelet features are concatenated 100-bin histograms of squared row norms of spectral filter operators, using PyGSP's Abspline filter bank with parameter 12; their kernel has $\sigma = 1$. Let D_j denote one of the five distances, and set

$$b_j = \mathrm{round}_4\, D_j(\mathcal{G}_{\mathrm{train}}, \mathcal{G}_{\mathrm{test}}), \quad J = \{j : b_j \neq 0\}, \qquad \mathrm{Ratio} = \frac{1}{|J|}\sum_{j\in J}\frac{D_j(\mathcal{G}_{\mathrm{gen}}, \mathcal{G}_{\mathrm{test}})}{b_j}.$$

Denominators are rounded to four decimal places; zeros are omitted. If all are zero, the returned sentinel -1 is excluded from comparison. Neither benchmark uses reference-size matching. Checkpoint selection maximizes validation V.U.N., breaking ties with a Ratio proxy using degree, clustering, and spectral statistics. Final test Ratio uses all five.

C.8 Dataset Configurations

Tables 7 and 8 list the KFM settings, each run with five independent seeds. Both stages train for 2,000 epochs, with flow condition dropout $p_{\mathrm{drop}} = 0.1$ for classifier-free guidance. Sampling uses Heun

Table 7: KFM model configurations. d is latent dimension and d_c is total condition dimension. H/L_E are encoder width/depth, H_D is decoder slot width, H_F/L_F are DiT width/depth, h_F is attention head count, and r_{PE} is the number of Laplacian feature channels.

Dataset	d	d_c	H	L_E	H_D	H_F	L_F	h_F	r_{PE}
Community	40	64	160	4	96	256	4	8	8
Ego	112	64	256	4	160	320	6	4	8
Grid	64	96	256	4	224	384	5	8	12
MUTAG	48	64	192	4	96	256	3	4	8
IMDB-B	32	64	128	4	128	384	6	8	8
Community-small	52	64	208	4	96	288	4	8	8
Ego-small	88	64	256	4	128	256	3	8	8
Planar	56	96	224	6	160	384	8	8	20
Tree	64	96	256	6	160	288	5	4	20

Table 8: KFM training and sampling configurations. α_{AE} and α_{FM} are the initial learning rates for the autoencoder and flow, B is batch size for both stages, p_{AE} is autoencoder dropout, σ_0 is base scale, w is guidance weight, and K is the number of integration steps.

Dataset	α_{AE}	α_{FM}	B	p_{AE}	σ_0	w	K
Community	2×10^{-4}	1.500×10^{-4}	160	0.100	1.000	1.100	360
Ego	5×10^{-4}	1.100×10^{-4}	224	0.080	0.800	1.100	280
Grid	2×10^{-4}	2×10^{-5}	12	0.050	1.000	1.150	160
MUTAG	2×10^{-4}	5×10^{-5}	40	0.050	1.000	1.050	120
IMDB-B	2.500×10^{-4}	1×10^{-4}	128	0.030	1.300	1.200	220
Community-small	2.500×10^{-4}	4×10^{-5}	44	0.100	0.900	1.050	100
Ego-small	4×10^{-4}	1×10^{-5}	24	0.050	1.000	1.050	80
Planar	2.500×10^{-4}	4×10^{-5}	20	0.050	1.000	1.150	120
Tree	3×10^{-4}	3×10^{-5}	16	0.050	1.000	1.150	200

integration. All datasets use $d_{\mathrm{learn}} = 64$; Grid, Planar, and Tree append $d_s = 32$ structural channels, giving $d_c = 96$ instead of 64. Encoder width follows $H = \max(128, \min(256, 4d))$. Ordinary BFS is enabled for Grid, MUTAG, and IMDB-B and disabled for Community, Ego, Community-small, and Ego-small. Planar/Tree use structural BFS (Appendix C.1).

Table 9 lists structural settings. All three datasets use edge-budget decoding and connectivity repair, with AE/flow validation every 100/200 epochs. Planar/Tree use histogram budget calibration at temperature 0.5 and the constrained decoding and candidate ranking in Appendix C.6.

D Additional Experiments

D.1 Clipping Diagnostics

We evaluate preprocessing round trips with one frozen diagnostic autoencoder per dataset (AE seed 1432), separate from the checkpoints used for generation results. The audit uses each model's original split, training-derived standardization statistics, base scale, and float32 geometry, including radius cap 0.996 and standard-deviation floor 10^{-4}, without retraining.

For each graph, we decode its original latent z_i and round-trip latent $\widetilde{z}_i$ after standardization and inverse preprocessing, including all projections. Both evaluations fix the condition, target adjacency, ordering, and node mask, with the decoder in evaluation mode. Edge F1 (threshold 0.5) and logit-based binary cross-entropy (BCE) use valid unordered node pairs before graph cleanup or constraints, excluding diagonal and padded entries. Changes are round-trip minus original, averaged across graphs. Relative error in the original tangent space is

$$e_i = \frac{\|\log_0^K(\widetilde{z}_i) - \log_0^K(z_i)\|_2}{\|\log_0^K(z_i)\|_2}.$$

Table 9: Additional settings for the structural pipeline. λ_{deg} and λ_{hist} weight the degree and soft-histogram losses in Eq. (18). The profile blend is the weight of the sampled degree profile in the decoding target; score noise is the construction-score perturbation scale. A dash marks a setting not used by the Grid decoding procedure.

Dataset	λ_{deg}	λ_{hist}	Profile blend	Score noise	Candidate multiplier
Grid	0.150	0.100	–	–	4
Planar	0.200	0.025	0.250	0.025	8
Tree	0.150	0.100	0.625	0.050	8

Table 10: Preprocessing diagnostics over all test graphs, including those without clipping, using one diagnostic AE per dataset. $\overline{e}$ is mean relative tangent-space error; mean F1 and BCE changes are reported in percentage points (pp) and 10^{-3}, respectively.

Dataset	N	Forward clip (%)	$\overline{e}$	Mean ΔF1 (pp)	Mean ΔBCE (10^{-3})
Community	200	76.500	0.380	-0.061	$+1.149$
IMDB-B	200	75.000	0.275	$+0.045$	$+0.676$
Ego	240	100.000	0.636	$+0.035$	-0.047
Planar	40	47.500	0.236	$+0.013$	-0.012
Tree	40	100.000	0.155	0.000	$+0.008$
Grid	16	62.500	0.254	-0.016	$+0.003$
MUTAG	38	100.000	0.490	-0.039	$+0.575$
Ego-small	40	100.000	0.570	-0.052	-5.038
Community-small	17	100.000	0.406	0.000	-0.186

Forward clipping measures projection of standardized targets mapped to the Klein ball, not projection along generated trajectories.

Across nine datasets, absolute mean F1 changes are at most 0.061 pp at the displayed precision (Table 10). Forward target clipping occurs on 47.5–100% of test graphs, with mean relative tangent-space errors of 0.155–0.636. All $100{,}000$ audited base samples per dataset are clipped under the recorded seed and scale, so projection actively shapes the implemented base distribution. These single-seed results show small average reconstruction changes despite latent distortion; per-graph effects vary, and small test sets limit conclusions about rare degradation. The audit does not assess generation MMD or sampling-trajectory accuracy. Only test results are reported; Community/Ego validation sets overlap AE training data and are not independent evidence.

D.2 Community-small and Ego-small

Table 11 compares degree, clustering, and orbit MMD on Community-small and Ego-small. Orbit features capture local graphlets (You et al., 2018); Appendix C.7 gives the kernels and preprocessing. KFM uses the standard pipeline and five-seed means. Baselines come from Table 2 of Bu et al. (2026), whose HVQVAE+Flow model is labeled GGBall. Space notation follows that source: $\mathbb{G}$ denotes graphs, $\mathbb{R}$ Euclidean space, $\mathbb{B}$ the Poincaré ball, and $\mathbb{K}$ the Klein ball (curvature -1 for KFM). These datasets use different kernels from the five main benchmarks and do not apply reference-size matching. GGBall combines a hyperbolic autoencoder with a Riemannian flow prior (Bu et al., 2026). KFM has higher average MMD than GGBall on both datasets, but lower degree MMD on Ego-small. At the displayed precision, KFM ranks third in average MMD on Community-small, ties for second on Ego-small, and ranks first in Ego-small degree MMD. The corresponding GGBall configuration and provenance are documented in Appendix D.5.

D.3 Backbone and Coordinate-representation Ablations

Table 12 extends the backbone and coordinate ablations to Community-small and Ego-small. At the displayed precision, DiT has lower mean MMD than the approximately parameter-matched U-Net in five of six comparisons; U-Net has lower degree MMD on Community-small. Klein has lower means than the Poincaré variant in five comparisons and ties on Community-small degree MMD. As

Table 11: MMD on Community-small and Ego-small (↓). Baselines are quoted from Bu et al. (2026); KFM reports five-training-seed means. Values are rounded to three decimal places. Average is computed from degree, clustering, and orbit MMD before rounding. **Bold**, underline, and *italics* denote the three best distinct displayed values; ties share a rank.

Type	Space	Method	Community-small				Ego-small			
			Degree	Clustering	Orbit	Average	Degree	Clustering	Orbit	Average
One-shot	$\mathbb{R}^d$	GraphVAE	0.350	0.980	0.540	0.623	0.130	0.170	0.050	0.117
Autoregressive	$\mathbb{G}^d$	GraphRNN	0.080	0.120	0.040	0.080	0.090	0.220	*0.003*	0.104
	$\mathbb{R}^d$	VQGAE	0.032	0.062	0.005	0.033	0.021	0.041	0.007	0.023
Diffusion	$\mathbb{G}^d$	EDP-GNN	0.053	0.144	0.026	0.074	0.052	0.093	0.007	0.051
	$\mathbb{G}^d$	GDSS	0.045	0.086	*0.007*	0.046	0.021	*0.024*	0.007	0.017
	$\mathbb{G}^d$	DiGress	0.047	**0.041**	0.026	0.038	–	–	–	–
	$\mathbb{G}^d \times \mathbb{B}^n$	HGDM	*0.017*	0.050	0.005	0.024	*0.015*	0.023	*0.003*	0.014
Flow	$\mathbb{G}^d$	GNF	0.200	0.200	0.110	0.170	0.030	0.100	**0.001**	0.044
	$\mathbb{G}^d$	GraphAF	0.060	0.100	0.015	0.058	0.040	0.040	0.008	0.029
	$\mathbb{G}^d$	CatFlow	0.018	0.086	*0.007*	0.037	0.013	*0.024*	0.008	*0.015*
	$\mathbb{B}_c^n$	GGBall	**0.002**	*0.059*	**0.004**	**0.022**	0.013	**0.018**	0.002	**0.011**
	$\mathbb{K}^d$	**KFM (ours)**	0.007	0.075	*0.007*	*0.030*	**0.004**	0.035	0.004	0.014

Table 12: Backbone and geometry ablations on Community-small and Ego-small. Lower MMD is better. All entries report mean ± sample standard deviation over five training seeds, rounded to three decimal places. Bold marks the lowest displayed mean within each dataset and metric.

Dataset	Variant	Degree↓	Clustering↓	Orbit↓
Community-small	Klein + DiT	0.007 ± 0.005	$\mathbf{0.075 \pm 0.030}$	$\mathbf{0.007 \pm 0.007}$
	Klein + U-Net	$\mathbf{0.006 \pm 0.002}$	0.106 ± 0.019	0.029 ± 0.021
	Poincaré + DiT	0.007 ± 0.006	0.118 ± 0.024	0.032 ± 0.003
Ego-small	Klein + DiT	$\mathbf{0.004 \pm 0.003}$	$\mathbf{0.035 \pm 0.021}$	$\mathbf{0.004 \pm 0.004}$
	Klein + U-Net	0.031 ± 0.017	0.097 ± 0.042	0.008 ± 0.005
	Poincaré + DiT	0.017 ± 0.011	0.066 ± 0.004	0.009 ± 0.004

in Section 4.3, the coordinate ablation reuses the frozen Klein VAE encoder–decoder and changes only the flow coordinates through isometric Klein–Poincaré transformations. Both ablations use the baseline node ordering throughout encoding, flow learning, decoding, and evaluation.

D.4 What Does the Flow Add?

Both the flow and frozen decoder receive graph-dependent conditions, and the structural pipeline also uses training-derived sizes and degree profiles. To assess the contribution of learned transport within this pipeline, we compare KFM with two no-flow latent controls and, on Planar and Tree, a profile-only construction control.

For each dataset, the two no-flow controls use the same frozen checkpoint. For every inference seed, we sample training-bank indices once and reuse the resulting condition/profile rows for both controls. Given an index $i \sim \mathcal{U}\{1, \dots, M\}$, their decoded latents are

$$z_{\text{post}} = \exp_0^K(u), \qquad u \sim q_\phi(u \mid G_i),$$
$$z_{\text{prior}} = \exp_0^K(u), \qquad u \sim \mathcal{N}(0, I).$$

Each latent is decoded as $D_\psi(z, c_i)$. The posterior control uses the instance-specific training-graph posterior; the VAE-prior latent is independent of that posterior. Both retain c_i because the decoder is conditional. On Planar and Tree, the profile-only control uses the same condition and training-derived structural profile but replaces learned construction scores with seeded random scores, retaining the constraints and candidate selection. This control measures the contribution of profiles and constrained construction.

Table 13: KFM and no-flow controls on Community-small and Ego-small: MMD (↓), rounded to three decimal places. Entries are mean ± sample standard deviation. KFM uses five training seeds, as in Table 12; controls use ten inference seeds from one frozen checkpoint. Bold marks the lowest displayed mean, including ties.

Dataset	Method	Degree↓	Clustering↓	Orbit↓
Community-small	KFM	$\mathbf{0.007 \pm 0.005}$	$\mathbf{0.075 \pm 0.030}$	$\mathbf{0.007 \pm 0.007}$
	AE posterior	0.012 ± 0.013	0.114 ± 0.052	0.028 ± 0.032
	VAE prior	0.011 ± 0.013	0.116 ± 0.052	0.028 ± 0.030
Ego-small	KFM	$\mathbf{0.004 \pm 0.003}$	$\mathbf{0.035 \pm 0.021}$	$\mathbf{0.004 \pm 0.004}$
	AE posterior	0.023 ± 0.012	0.096 ± 0.030	0.006 ± 0.004
	VAE prior	0.023 ± 0.012	0.095 ± 0.030	0.006 ± 0.004

The control runs retain the same numerical Klein operations, condition noise, decoder, graph cleanup or constrained construction, candidate-pool size, candidate-selection rule, generated sample count, held-out reference split, and evaluator used by the full KFM protocol. For each dataset, we use one frozen checkpoint trained under the same configuration as the corresponding main experiment and evaluate it only on the held-out test split using ten inference seeds. Both controls use the same ten inference seeds and shared condition/profile draws; these seeds measure sampling variability and are not independent training runs. We report mean ± sample standard deviation.

Tables 5 and 6 report the five main datasets and constrained-generation benchmarks; Table 13 reports the small datasets. KFM entries reuse the means over five independent training seeds. Its standard deviations include training and sampling variation, whereas the controls' standard deviations measure sampling variation from a single frozen checkpoint.

Across the five main datasets and two small datasets, the displayed KFM mean is lower than the AE-posterior mean in 20 of 21 dataset–statistic entries; degree MMD on Grid is the only exception. Relative to VAE-prior sampling, KFM is lower in 17 of 21 entries and higher on Grid degree. Grid clustering and MUTAG degree and clustering are tied at three decimal places. These results support a contribution of learned transport within the conditional pipeline. Because all methods retain training-derived conditions and the posterior control additionally accesses instance-specific latents, the comparison is a diagnostic of this pipeline, not an evaluation of standalone unconditional generators.

D.5 GGBall Baseline Configuration and Extended Evaluation

GGBall (Bu et al., 2026) combines a hyperbolic graph encoder, vector quantization, a pairwise decoder, and a flow prior in the Poincaré ball. It differs from KFM in its representation model, decoder, latent transport, and coordinate system. We therefore treat the comparison as a full-architecture comparison rather than as an isolated comparison between Poincaré and Klein coordinates.

On Community-small and Ego-small, Table 11 quotes the HVQVAE+Flow results from Table 2 of Bu et al. (2026), labeled GGBall here. These published results retain their original evaluation protocol and are distinct from our reproductions on the five main datasets.

For Community, Ego, Grid, MUTAG, and IMDB-B, we reproduce the publicly released GGBall implementation using the same data splits, preprocessing, MMD kernels, and reference-node-count matching as KFM (Appendix C.7). We report means over five independently trained seeds. These experiments provide an additional full-system comparison for Table 1.

Shared GGBall configuration. All reproductions use the HVQVAE architecture on the Poincaré ball with no Euclidean latent channels, curvature 0.6, a two-layer GNN encoder, a four-head distance-attention transformer with hidden dimension 16 and middle dimension 64, positional encoding, and a pairwise decoder. We optimize the VAE using AdamW with weight decay 10^{-12} and gradient clipping at 1.0. The VQ loss weight is 1.0, the orthogonal-regularization weight is 0.1, and the default codebook size is 32. Dataset-specific settings are reported in Table 14.

Table 14: Dataset-specific GGBall HVQVAE configurations used in the five-seed reproduction.

Dataset	Codebook	Hyp. dim.	Enc. Trans. layers	Dropout	VAE batch	Flow batch	λ_{band}	λ_{commit}
Community	32	64	8	0.10	256	128	1.0	2.0
Ego	32	64	8	0.10	128	64	1.0	2.0
Grid	16	64	4	0.10	32	32	0.1	1.0
MUTAG	32	32	4	0.15	16	32	0.3	1.0
IMDB-B	32	64	4	0.15	16	32	0.3	1.0

Extended evaluation. Table 15 reports reconstruction and generation metrics separately. Reconstruction edge accuracy measures edges decoded from observed-graph latent representations. Degree, clustering, spectral, and orbit MMD instead compare the distributions of statistics computed from generated and reference graphs. These quantities evaluate different stages of the pipeline and should not be interpreted interchangeably.

Table 15: GGBall reproduction on the five main datasets using the KFM data-processing and evaluation protocol. Values are means over five independent training seeds. Higher reconstruction edge accuracy and lower generation MMD are better. A dash denotes an unavailable value.

	Recon. Edge	Generation MMD ↓			
Dataset	Acc. (%) ↑	Degree	Clustering	Spectral	Orbit
Community	86.5	0.137	0.173	–	0.707
Ego	56.2	0.412	0.429	0.204	0.592
Grid	91.7	0.252	0.872	0.079	0.468
MUTAG	93.0	0.570	1.000	0.143	1.026
IMDB-B	65.7	0.110	0.071	0.109	–

Among the degree, clustering, and spectral entries shared with Table 1, KFM obtains a lower mean MMD than the reproduced GGBall baseline except on Grid spectral MMD. Community spectral MMD is unavailable for GGBall. Reconstruction accuracy and orbit MMD are reported as additional diagnostics and are not included in this comparison. Because GGBall and KFM differ in their autoencoders, latent representations, decoders, and flow parameterizations, these results should be interpreted as a comparison between the complete pipelines rather than as evidence that one hyperbolic coordinate system is intrinsically superior to the other.