GRAPHONFLOW: LEARNING LATENT GRAPHON DISTRIBUTIONS FOR STRUCTURE-PRESERVING GRAPH GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph generative models must capture complex structural dependencies while preserving domain-specific constraints. However, existing flow-based approaches typically initialise generation from independent noise or fixed priors, leaving the flow model to recover much of the target structure during generation. We propose *GraphonFlow*, a probabilistic framework that learns a latent-conditioned structured source distribution for flow-based graph generation. Our key idea is to integrate latent graph representation learning with graphon modelling, providing an informative structural starting point for subsequent flow learning. A two-stage training strategy first learns the latent representation and graphon-based source, and then freezes this source while learning a conditional velocity field that transforms structured source samples toward target graphs. To support structure-preserving generation, *GraphonFlow* further incorporates canonical graph representations for consistent graphon learning and connectivity-aware regularisation across the generative process. Experiments on four molecular benchmarks and three synthetic graph families demonstrate competitive performance in chemical validity, diversity, and structural fidelity. Extensive ablations further demonstrate the importance of the learned graphon source, canonical representation, two-stage optimisation, and connectivity-aware learning. Overall, our results show that learning the source distribution itself provides an effective alternative to initialising graph flows from unstructured or fixed priors.

1 INTRODUCTION

Learning generative models of graphs is a fundamental problem in machine learning with broad applications spanning molecular discovery, protein engineering, biological interaction networks, and social systems (Zhu et al., 2022; Jin et al., 2018; Yi et al., 2023). Successful graph generation must simultaneously preserve global topology, higher-order structural organisation, and domain-specific constraints such as chemical valence (Zang & Wang, 2020; Wijesinghe et al., 2026) or community and hub structure (Ravasz et al., 2002; Watts & Strogatz, 1998), while producing diverse, previously unseen instances.

Recent advances in deep graph generation span autoregressive models, VAEs, diffusion models, and continuous-time flow matching (Kipf & Welling, 2016; Jin et al., 2018; Lipman et al., 2023; Liu et al., 2023), each with complementary limitations: autoregressive models sacrifice permutation invariance to node ordering (You et al., 2018; Kong et al., 2023); latent-variable models restore permutation symmetry but struggle to reconstruct higher-order motifs from a compressed latent space (Kipf & Welling, 2016; Jin et al., 2018); diffusion models achieve strong empirical performance but model graphs only implicitly through stochastic corruption, requiring many denoising steps (Niu et al., 2020; Vignac et al., 2023). Flow matching instead learns deterministic continuous-time transport for faster sampling (Lipman et al., 2023; Liu et al., 2023; Qin et al., 2025), but existing graph flow models typically initialise transport from simple or manually specified source priors, forcing the velocity field to simultaneously recover global topology and refine local attributes.

A complementary line of work models graph distributions through *graphons* (Glasscock, 2016; Diaconis & Janson, 2007): symmetric measurable functions $W : [0, 1]^2 \to [0, 1]$ that define edge

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.

probabilities directly in a latent node-coordinate space, representing recurring global patterns such as communities and hubs through the geometry of a probability surface. Recent work has begun using graphons and related structured objects as priors for graph representation learning and generation (Xu et al., 2021; Xia et al., 2023; Wijesinghe et al., 2026). However, existing approaches either estimate graphons independently of a latent representation or use fixed structural priors (Xu et al., 2021; Xia et al., 2023; Wijesinghe et al., 2026), and graphons are further complicated by non-identifiability under measure-preserving relabelling, which we resolve via canonical graph labelling (Appendix E.1). A broader discussion of the relevant literature is provided in Appendix F.

This motivates learning the structural source itself from data, rather than specifying it independently of the generative process. To this end, we introduce *GraphonFlow*, a unified probabilistic framework that integrates latent graph representation learning, graphon-based structural priors, and continuous-time transport through a two-stage architecture: it first learns latent-conditioned structural priors and then uses the resulting graphon distributions as structured sources for continuous-time transport, in contrast to approaches that estimate graphons independently or employ fixed structural priors. As illustrated in Figure 1, *GraphonFlow* follows a two-stage architecture in which a graph variational autoencoder learns a latent-conditioned graphon that serves as both the Bernoulli adjacency decoder and the structured source for subsequent conditional flow matching. This coupling allows the transport process to refine structured graphs rather than construct topology from unstructured noise. *GraphonFlow* introduces three key ideas. First, a hybrid latent representation combines a graph-level latent capturing global topology with permutation-equivariant node-level latents preserving local structural and attribute information. Second, canonical graphon supervision maps isomorphic graphs to a unique canonical representation via McKay's NAUTY algorithm with property-aware vertex colouring (McKay, 1981), ensuring equivalent graph structures induce identical graphon targets without requiring graph matching, optimal-transport alignment, or additional learnable alignment modules. Third, connectivity-aware learning combines an Erdős–Rényi-inspired graphon regulariser (Newman, 2002), which promotes connected graphon-derived sources, with a differentiable, Lanczos-accelerated spectral regulariser based on algebraic connectivity that promotes connectivity during continuous-time transport, together encouraging structurally coherent and connected graphs throughout generation.

2 Background

2.1 Permutation Equivalence and Canonical Graph Labelling

A graph with n nodes is represented as $G = (A, X, E)$, with adjacency $A \in \{0,1\}^{n\times n}$, node attributes $X \in \mathbb{R}^{n\times d_x}$, and edge attributes $E \in \mathbb{R}^{n\times n\times d_e}$, for undirected attributed graphs ($A = A^\top$, $E_{ij} = E_{ji}$). A fixed matrix representation is not permutation invariant: two isomorphic graphs $G \cong G'$ are related by a permutation matrix P acting jointly on (A, X, E) (Appendix E.1, Eq. 83), so the same graph can be represented by different adjacency and feature matrices, which is particularly problematic for graphon learning, where each matrix position (i, j) corresponds to a fixed location on a continuous graphon surface. We eliminate this ambiguity via a canonical graph representation: a deterministic mapping $C : \mathcal{G} \to \mathcal{G}$ over the space of graphs $\mathcal{G}$, satisfying $G \cong G' \Leftrightarrow C(G) = C(G')$, computed using McKay's NAUTY algorithm (McKay, 1981) with optional vertex colouring for domain attributes (e.g., atom type and formal charge). Full algorithmic details, the resulting combinatorial reduction, and the role of canonicalisation in eliminating permutation ambiguity from graphon supervision are given in Appendix E.1.

2.2 Flow Matching

Flow Matching (FM) (Lipman et al., 2023; Liu et al., 2023) learns a deterministic transport map between a source distribution p_0 and a target distribution p_{data} over graphs. Given a source graph $G_0 \sim p_0$ and a target graph $G_1 \sim p_{\text{data}}$, FM constructs a probability path through the rectified linear interpolation

$$G_t = (1-t)G_0 + tG_1, \qquad t \in [0,1], \tag{1}$$

with corresponding target velocity $u_t(G_t) = dG_t/dt = G_1 - G_0$. A neural network v_θ is trained to approximate this velocity field by regression, and generation proceeds by integrating the resulting ODE from $t = 0$ to $t = 1$. A full exposition, including the FM regression objective and its dependence on the source distribution p_0, is given in Appendix E.2.

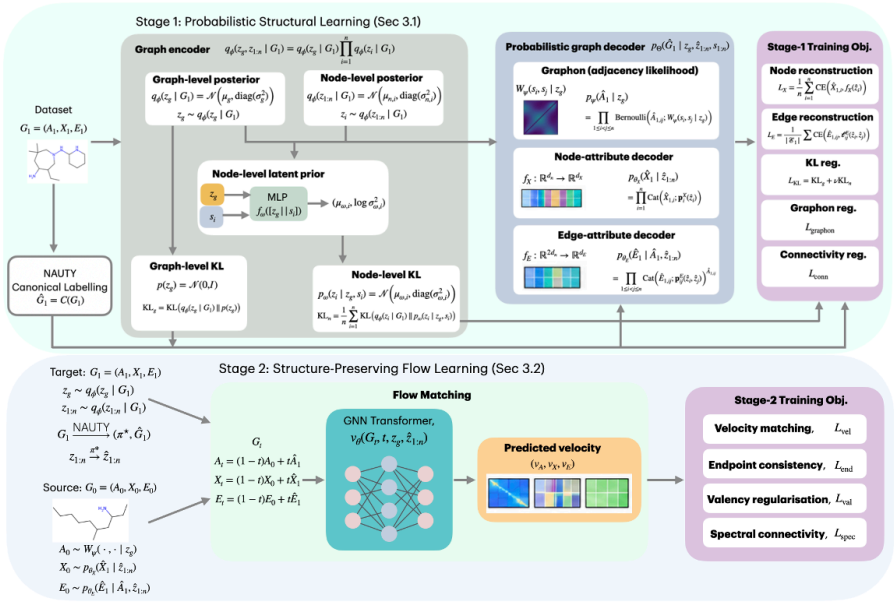


Figure 1: Overview of GraphonFlow. Stage 1 (Sec. 3.1) learns a latent-conditioned graphon and probabilistic graph decoder from canonically aligned target graphs. Stage 2 (Sec. 3.2) freezes the learned components and uses them to construct a structured source G_0, which is transported toward the canonical target $\hat{G}_1$ by a latent-conditioned GNN Transformer. Connectivity regularisation is applied in both stages (Sec. 3.3).

2.3 Graphons as Structural Graph Priors

Graphons provide a principled probabilistic representation of graph distributions and constitute the infinite-dimensional limit of large exchangeable graphs (Lovász, 2012). Formally, a graphon is a symmetric measurable function $W : [0,1]^2 \to [0,1]$, where $W(x,y)$ denotes the probability that two latent nodes at positions x, y are connected. Given W, a graph with n nodes is generated by sampling latent coordinates $s_1, \ldots, s_n \overset{\text{i.i.d.}}{\sim} \text{Unif}[0,1]$ and edges according to

$$A_{ij} \sim \text{Bernoulli}(W(s_i, s_j)), \qquad 1 \le i < j \le n. \tag{2}$$

Graphon parameterisation, size-dependent sparsification, and related structural graph priors are discussed in Appendix E.3; our latent-conditioned construction is introduced in Section 3.1.

2.4 Algebraic Graph Connectivity

We use algebraic connectivity as a differentiable measure for promoting connected graph structures during transport. Full background and derivations are provided in Appendix E.4; its use in *GraphonFlow* is introduced in Section 3.3.

3 Latent-Conditioned Graphon Flow Matching

Figure 1 summarises the two-stage architecture of *GraphonFlow*. Stage 1 learns a probabilistic structural model in which graph- and node-level latent representations parameterise a latent-conditioned graphon and attribute decoders under canonical supervision. Stage 2 freezes these components and reuses them to construct a structured source graph G_0 and condition flow matching toward the canonical target $\hat{G}_1$. The flow operates directly on the evolving graph state $G_t = (A_t, X_t, E_t)$, while the learned latent representations remain fixed conditioning variables. Connectivity-aware regularisation is incorporated in both stages. Section 3.1 describes the Stage 1 probabilistic structural model, Section 3.2 presents the Stage 2 structure-preserving flow, and Section 3.3 introduces the connectivity-aware regularisation used across both stages.

3.1 Stage 1: Probabilistic Structural Learning

Stage 1 learns the latent representation, latent-conditioned graphon, and probabilistic graph decoder jointly. *GraphonFlow* decomposes the latent representation into complementary graph- and node-level components. The graph-level latent $z_g \in \mathbb{R}^{d_g}$ parameterises the global graphon surface and captures connectivity structure, while node-level latents $z_{1:n} \in \mathbb{R}^{n \times d_n}$ encode local information for node- and edge-attribute reconstruction. Edge existence is decoded solely through the graphon; directly conditioning it on node-level latents would instead reduce the structural decoder toward pairwise link prediction, weakening the graphon's role as a global structural prior. This design is discussed further in Appendix G.3.

Graph encoder and variational posterior. Given an input graph $G_1 = (A_1, X_1, E_1)$ in its observed node ordering, the encoder q_ϕ produces graph- and node-level posteriors

$$q_\phi(z_g, z_{1:n} \mid G_1) = q_\phi(z_g \mid G_1) \prod_{i=1}^{n} q_\phi(z_i \mid G_1), \tag{3}$$

with

$$q_\phi(z_g \mid G_1) = \mathcal{N}\big(\mu_g, \mathrm{diag}(\sigma_g^2)\big), \qquad q_\phi(z_i \mid G_1) = \mathcal{N}\big(\mu_{n,i}, \mathrm{diag}(\sigma_{n,i}^2)\big). \tag{4}$$

The graph-level posterior is permutation invariant through the graph readout, whereas the collection of node-level posterior parameters is permutation equivariant. The posterior parameters $(\mu_g, \log \sigma_g^2)$ and $(\mu_{n,i}, \log \sigma_{n,i}^2)$ are read out from an edge-aware GNN encoder, and latent variables are sampled using the standard reparameterisation estimator. Full encoder and sampling details are provided in Appendix G.1.

Canonical graph representation. Graph reconstruction is performed against a canonical representative rather than an arbitrary node ordering. The encoder first processes G_1 in its observed node ordering. Independently, we compute a canonical labelling using NAUTY, yielding a permutation $\pi^\star$ and the canonical target

$$\hat{G}_1 = C(G_1) = (\hat{A}_1, \hat{X}_1, \hat{E}_1),$$

where

$$\hat{A}_1 = A_{1,\pi^\star,\pi^\star}, \qquad \hat{X}_1 = X_{1,\pi^\star}, \qquad \hat{E}_1 = E_{1,\pi^\star,\pi^\star}.$$

The same permutation is applied to the sampled node-level latent representations,

$$\hat{z}_i = z_{\pi^\star(i)}.$$

The graph-level latent z_g is unaffected by this reordering because the graph-level readout is permutation invariant. Consequently, graphon, node-attribute, and edge-attribute reconstruction are evaluated in a common canonical coordinate system.

Generative model and conditional node prior. Let $s_{1:n} = \{s_i\}_{i=1}^n$, with $s_i = (i-1)/(n-1) \in [0,1]$, denote deterministic canonical lattice coordinates. *GraphonFlow* defines

$$p_\Theta(\hat{G}_1, z_g, \hat{z}_{1:n} \mid s_{1:n}) = p(z_g)\, p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n})\, p_\Theta(\hat{G}_1 \mid z_g, \hat{z}_{1:n}, s_{1:n}), \tag{5}$$

where $p(z_g) = \mathcal{N}(0, I)$. Conditioned on z_g and the canonical node coordinates,

$$p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n}) = \prod_{i=1}^{n} p_\omega(\hat{z}_i \mid z_g, s_i), \qquad p_\omega(\hat{z}_i \mid z_g, s_i) = \mathcal{N}\big(\mu_{\omega,i}, \mathrm{diag}(\sigma_{\omega,i}^2)\big), \tag{6}$$

with $(\mu_{\omega,i}, \log \sigma_{\omega,i}^2) = f_\omega([z_g \| s_i])$. Thus z_g provides global context shared across the graph, while s_i identifies the corresponding canonical node position. Despite this position-dependent conditioning, the induced likelihood on unlabeled graphs is permutation invariant (Proposition 4, Appendix C).

Latent-conditioned graphon. The graph-level latent representation z_g conditions a graphon that captures the global connectivity patterns associated with the encoded graph. We first define the unsparsified graphon

$$\bar{W}_\psi(s_i, s_j \mid z_g) = \sigma\left(\sum_{b \in \mathcal{B}} \theta_b(z_g) f_b(s_i, s_j)\right), \tag{7}$$

and obtain the effective graphon

$$W_\psi(s_i, s_j \mid z_g) = \rho(z_g)\, \bar{W}_\psi(s_i, s_j \mid z_g), \tag{8}$$

where $\{f_b\}_{b\in\mathcal{B}}$ denotes a symmetric graphon basis, $\theta(z_g)$ the corresponding basis coefficients, and $\rho(z_g) \in (0, 1)$ a latent-conditioned sparsification factor. Both $\theta(z_g)$ and $\rho(z_g)$ are predicted jointly by $g_\psi : \mathbb{R}^{d_g} \to \mathbb{R}^{|\mathcal{B}|+1}$. Thus z_g specifies which graphon is used, while the canonical lattice coordinates determine where on that graphon each edge probability is evaluated.

GraphonFlow supports Fourier and Bernstein parameterisations. Full basis definitions, approximation properties, and convergence guarantees are provided in Appendices A and B.

Probabilistic graph decoder. The final factor in Eq. 5 specifies the conditional distribution of the canonical graph. It factorises as

$$p_\Theta(\hat{G}_1 \mid z_g, \hat{z}_{1:n}, s_{1:n}) = p_\psi(\hat{A}_1 \mid z_g, s_{1:n})\, p_{\theta_X}(\hat{X}_1 \mid \hat{z}_{1:n})\, p_{\theta_E}(\hat{E}_1 \mid \hat{A}_1, \hat{z}_{1:n}). \tag{9}$$

The adjacency likelihood is

$$p_\psi(\hat{A}_1 \mid z_g) = \prod_{1\le i<j\le n} \text{Bernoulli}\Big(\hat{A}_{1,ij}; W_\psi(s_i, s_j \mid z_g)\Big), \tag{10}$$

while node and edge attributes are reconstructed from the canonically reordered node latents using categorical decoders f_X and f_E. The pairwise edge decoder is symmetrised to preserve undirectedness. Complete attribute likelihoods and the corresponding reconstruction losses $\mathcal{L}_X$ and $\mathcal{L}_E$ are given in Appendix G.2.

Accordingly, adjacency reconstruction is performed using the binary cross-entropy induced by the graphon likelihood,

$$\mathcal{L}_{\text{graphon}} = -\tfrac{1}{\binom{n}{2}} \textstyle\sum_{1\le i<j\le n} \Big[\hat{A}_{1,ij} \log W_\psi(s_i, s_j \mid z_g) + (1 - \hat{A}_{1,ij}) \log\big(1 - W_\psi(s_i, s_j \mid z_g)\big)\Big]. \tag{11}$$

Combining these, the Stage 1 reconstruction objective is, $\mathcal{L}_{\text{rec}} = \mathcal{L}_X + \mathcal{L}_E + \beta_g \mathcal{L}_{\text{graphon}}$, where β_g controls the relative contribution of graphon-based adjacency reconstruction; $\mathcal{L}_{\text{graphon}}$ is averaged over all $\binom{n}{2}$ canonical node pairs, $\mathcal{L}_X$ over the n canonical nodes, and $\mathcal{L}_E$ over the $|\mathcal{E}_1|$ present canonical edges.

The canonical ordering provides the consistent correspondence between vertices and graphon coordinates required for coordinate-wise graphon supervision. Without canonicalisation, isomorphic orderings can assign inconsistent adjacency targets to the same graphon coordinates. Consequently, the composition of canonicalisation and graphon supervision is independent of the observed vertex ordering: isomorphic orderings are mapped to the same canonical representative and yield the same graphon loss and gradient with respect to ψ. This property is formalised in Theorem 3 (Appendix C).

3.2 Stage 2: Structure-Preserving Flow Learning

Once Stage 1 has learned the latent representation, graphon, and probabilistic decoders, these components are frozen and reused to define the source and latent conditioning for flow matching. Unlike graph flow models that initialise transport from simple or manually specified source priors, *GraphonFlow* begins from a graph whose topology and attributes are generated by the learned Stage 1 structural model.

Frozen encoder conditioning. For each training target G_1, the frozen Stage 1 encoder is reused to sample

$$z_g \sim q_\phi(z_g \mid G_1), \qquad z_i \sim q_\phi(z_i \mid G_1), \quad i = 1, \dots, n. \tag{12}$$

In parallel, the same NAUTY canonicalisation used during Stage 1 yields $\hat{G}_1 = C(G_1)$ and its permutation $\pi^\star$. The sampled node-level latents are placed into the same canonical order,

$$\hat{z}_i = z_{\pi^\star(i)}, \qquad i = 1, \dots, n, \tag{13}$$

while z_g is invariant to node permutations. The resulting $(z_g, \hat{z}_{1:n})$ are fixed conditioning variables throughout the transport process; they are not part of the evolving graph state.

Structured source construction. The frozen Stage 1 model uses these latent variables to construct $G_0 = (A_0, X_0, E_0)$. Its topology is sampled from the latent-conditioned graphon,

$$A_{0,ij} \sim \text{Bernoulli}(W_\psi(s_i, s_j \mid z_g)), \tag{14}$$

while X_0 and E_0 are generated from the canonically aligned node-level latents $\hat{z}_{1:n}$ using the frozen attribute decoders f_X and f_E, respectively. The resulting conditional source distribution can therefore be written as

$$G_0 \sim p_0(\cdot \mid z_g, \hat{z}_{1:n}, s_{1:n}).$$

The graph-level latent controls source connectivity through the graphon, whereas the node-level latents provide locally varying node and edge semantics.

Conditional flow matching in graph space. Given the structured source G_0 and canonical target $\hat{G}_1$, we define

$$G_t = (1-t)G_0 + t\hat{G}_1, \qquad t \sim \text{Unif}(0,1). \tag{15}$$

Equivalently,

$$A_t = (1-t)A_0 + t\hat{A}_1, \qquad X_t = (1-t)X_0 + t\hat{X}_1, \qquad E_t = (1-t)E_0 + t\hat{E}_1.$$

The flow therefore evolves the continuous graph state $G_t = (A_t, X_t, E_t)$; neither the graphon nor the latent representations are evolved during transport.

Latent-conditioned velocity field. The velocity field is implemented by an edge-aware GNN Transformer,

$$v_\theta(G_t, t, z_g, \hat{z}_{1:n}) = (v_A, v_X, v_E), \tag{16}$$

which receives the evolving graph state G_t together with time t and the fixed latent conditioning variables $(z_g, \hat{z}_{1:n})$. Through message passing and self-attention, it jointly predicts adjacency, node-attribute, and edge-attribute velocities, with adjacency and edge outputs symmetrised to preserve undirectedness. The complete architecture is provided in Appendix G.4.

3.3 Connectivity-Aware Learning

Connectivity regularisation complements the two-stage architecture at different points in the generative process. During Stage 1, the learned graphon is regularised to produce sufficiently connected structured sources. During Stage 2, a differentiable spectral objective is applied to the predicted endpoint to promote connectivity after transport.

Stage 1: Graphon connectivity regularisation. Our graphon defines a probability distribution over source graphs through $A_0 \sim \mathbb{G}(n, W_n)$, where $W_n(s_i, s_j \mid z_g) = \rho(z_g)\,\bar{W}_\psi(s_i, s_j \mid z_g)$ is the sparsified latent-conditioned graphon. Writing $\bar{W}_n(z_g)$ for the effective mean edge probability of this sparsified surface, we define

$$\mathcal{L}_{\text{conn}}(z_g, n) = \eta\left(\frac{1}{\bar{W}_n(z_g)\,n + \xi} - \rho(z_g)\right) + \eta\big(\delta - \bar{W}_n(z_g)\big), \tag{17}$$

where $\eta(\cdot)$ denotes the ReLU activation function, $\xi > 0$, and $\delta > 0$. The first term relates the sparsification factor to the effective graphon density, while the second prevents the source density from collapsing toward zero. The regulariser's derivation and estimator $\bar{W}_n(z_g)$ are provided in Appendix G.6, while the corresponding connectivity result is established in Theorem 5 (Appendix C).

Stage 2: Spectral connectivity regularisation. While the graphon regulariser encourages connected source graphs, continuous-time transport may still weaken connectivity as edge probabilities evolve. Given the predicted endpoint adjacency $\tilde{A}_1 = A_t + (1-t)v_A$ (Appendix G.5, Eq. 104), we define the regularised soft Laplacian $\tilde{L}_1 = \text{diag}(\tilde{A}_1\mathbf{1}) - \tilde{A}_1$ and encourage connectivity through its second-smallest eigenvalue,

$$\mathcal{L}_{\text{spec}} = \eta\Big(\gamma - \hat{\lambda}_2\Big), \tag{18}$$

where $\hat{\lambda}_2 \approx \lambda_2(\tilde{L}_1)$ is obtained through a differentiable deflated Lanczos approximation (Algorithm 1, Appendix D). Further details of the spectral regulariser are provided in Appendix G.6, while the

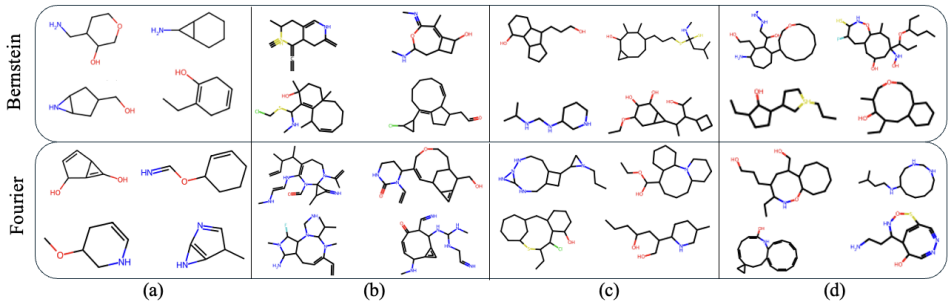


Figure 2: Molecular graphs generated by *GraphonFlow* across four benchmark datasets. Representative samples are shown for (a) QM9, (b) ZINC250K, (c) MOSES, and (d) Guacamol. The top and bottom rows correspond to the Bernstein and Fourier graphon parameterisations, respectively.

corresponding trajectory connectivity analysis and computational complexity are given in Appendix C and Appendix H, respectively.

Together, these objectives promote connected source graphs before transport and connected predicted endpoints after transport, complementing the structural information supplied by the learned graphon source.

4 Training

4.1 Two-Stage Optimisation

The preceding construction assigns two distinct roles to learning: estimating a stable latent-conditioned structural source and learning the dynamics that transport samples from this source to data graphs. We therefore optimise *GraphonFlow* in two stages, first learning the latent representation and structural source model, and then freezing these components while training the conditional velocity field.

4.2 Stage 1: Probabilistic Structural Learning

The objective of Stage 1 is to learn a latent graph representation together with a latent-conditioned graphon that reconstructs the canonical graph $\hat{G}_1 = (\hat{A}_1, \hat{X}_1, \hat{E}_1)$. Following the variational autoencoder (VAE) framework (Kingma & Welling, 2013), we maximise the Evidence Lower Bound (ELBO) on the intractable marginal likelihood $\log p_\Theta(\hat{G}_1)$. Its decomposition into a reconstruction term and two KL terms is derived in Theorem 9.

KL regularisation. The graph- and node-level posteriors are regularised against their respective priors through $\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}_g + \nu\, \mathbb{E}_{q_\phi(z_g|G_1)}\left[\mathrm{KL}_n(z_g)\right]$. Complete ELBO derivation, KL definitions, normalisation conventions, Monte Carlo estimation, and annealing details are provided in Appendix G.7.

GraphonFlow further employs two deterministic regularisers during Stage 1: a normalised ℓ_1 regulariser encouraging compact graphon parameterisations, $\mathcal{L}_{\ell_1} = \frac{1}{B}\sum_{b=1}^{B} |\theta_b(z_g)|$, where B is the number of Fourier or Bernstein basis coefficients; and the connectivity regulariser $\mathcal{L}_{\mathrm{conn}}$ (Section 3.3), which encourages the latent graphon to generate sufficiently connected source graphs before transport begins. Combining these components, the complete Stage 1 objective is

$$\mathcal{L}_{\mathrm{S1}} = \mathbb{E}_{G_1\sim\mathcal{D}}\Big[\mathbb{E}_{q_\phi(z_g,z_{1:n}|G_1)}\Big[\mathcal{L}_X + \mathcal{L}_E + \beta_{\mathrm{g}}\mathcal{L}_{\mathrm{graphon}} + \beta_{\ell_1}\mathcal{L}_{\ell_1} + \beta_{\mathrm{c}}\mathcal{L}_{\mathrm{conn}}(z_g, n)\Big] + \beta_{\mathrm{KL}}(t)\mathcal{L}_{\mathrm{KL}}\Big]. \tag{19}$$

4.3 Stage 2: Structure-Preserving Flow Learning

Once the latent representation and graphon prior have converged, their parameters are fixed and only the conditional velocity field is optimised. Freezing the learned graphon prevents the flow-matching objective from modifying the structured source distribution learned during Stage 1, allowing the

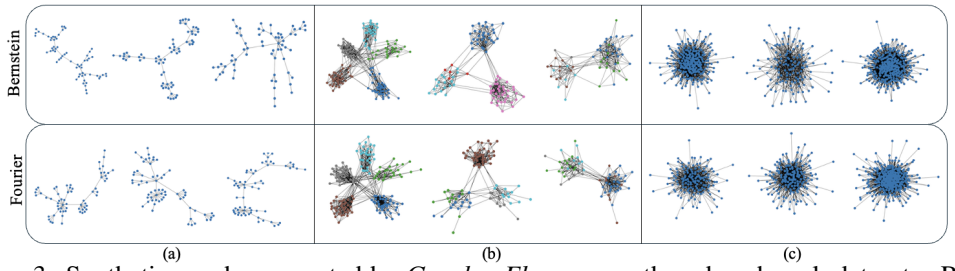


Figure 3: Synthetic graphs generated by *GraphonFlow* across three benchmark datasets. Representative samples are shown for (a) Tree, (b) SBM, and (c) Ego-small. The top and bottom rows correspond to the Bernstein and Fourier graphon parameterisations, respectively.

velocity field to focus on learning the transformation from structured source samples toward the corresponding target graphs.

For each target graph, latent variables are sampled using the frozen Stage 1 encoder, and the structured source G_0 is constructed from the learned graphon and attribute decoders as defined in Section 3.2. The complete training procedure is given in Algorithm 2. The Stage 2 objective is

$$\mathcal{L}_{\mathrm{S2}} = \mathbb{E}_{G_1\sim\mathcal{D}}\, \mathbb{E}_{G_0\sim p_0(\cdot|z_g,\hat{z}_{1:n},s_{1:n})}\, \mathbb{E}_{t\sim\mathrm{Unif}[0,1]} \left[\mathcal{L}_{\mathrm{vel}} + \beta_{\mathrm{end}}\mathcal{L}_{\mathrm{end}} + \beta_{\mathrm{val}}\mathcal{L}_{\mathrm{val}} + \beta_{\mathrm{spec}}\mathcal{L}_{\mathrm{spec}}\right], \quad (20)$$

where the velocity field is evaluated along the interpolated graph $G_t = (1-t)G_0 + t\hat{G}_1$, using the conditional velocity network $v_\theta(G_t, t, z_g, \hat{z}_{1:n})$. The complete definitions of the flow-matching loss terms are provided in Appendix G.5.

The velocity loss encourages accurate transport between source and target graphs, while the endpoint, valence, and spectral objectives promote accurate endpoint prediction, chemical validity, and connectivity during flow learning. Together, the two-stage optimisation first learns an expressive probabilistic structural prior and subsequently learns continuous graph transport conditioned on that prior, enabling the velocity field to refine realistic graph structures rather than reconstructing them from unstructured noise. The graph generation procedure using the resulting frozen model is described in Appendix G.8, and a detailed computational complexity analysis is provided in Appendix H.

5 Numerical Experiments

We evaluate *GraphonFlow* on synthetic and molecular benchmarks, assessing structural fidelity and molecular generative quality, respectively. Unlike methods relying on manually specified structural priors, *GraphonFlow* learns its latent-conditioned graphon directly from the training distribution. We evaluate both Fourier and Bernstein graphon parameterisations against established graph generative baselines.

5.1 Molecular Graph Generation

We assess *GraphonFlow* on four widely used molecular generation benchmarks: QM9 (Wu et al., 2018), ZINC250K (Sterling & Irwin, 2015), Guacamol (Brown et al., 2019), and MOSES (Polykovskiy et al., 2020). These datasets provide complementary evaluation regimes, ranging from small molecular graphs in QM9 to larger and more structurally diverse drug-like molecules in ZINC250K, Guacamol, and MOSES. For QM9 and ZINC250K, we adopt the standard data splits and generation metrics used by prior graph generative models (Vignac et al., 2023; Qin et al., 2025), reporting validity, uniqueness, and NSPDK. For Guacamol and MOSES, we follow their established benchmark protocols (Brown et al., 2019; Polykovskiy et al., 2020), which additionally assess KL divergence of physicochemical properties (KL div.), novelty, Similarity to Nearest Neighbour (SNN), and Scaffold similarity (Scaff.). Results for the two *GraphonFlow* parameterisations are reported in Tables 1 and 2. Appendix I.2 provides the complete experimental setup and discussion.

5.2 Synthetic Graph Generation

We evaluate *GraphonFlow* on three established synthetic graph benchmarks: Tree (Bergmeister et al., 2024), Stochastic Block Model (SBM) (Martinkus et al., 2022), and Ego-small (Jo et al., 2022).

Table 1: Generation performance on QM9 and ZINC250K molecular graphs.

Model	QM9			ZINC250K		
	Valid ↑	Unique ↑	NSPDK ↓	Valid ↑	Unique ↑	NSPDK ↓
EDP-GNN (Niu et al., 2020)	47.52	99.25	0.0046	82.97	99.79	0.0485
GraphARM (Kong et al., 2023)	90.25	95.65	0.002	88.23	99.46	0.055
DiGress (Vignac et al., 2023)	98.19	96.20	0.0003	94.99	99.97	0.0021
GDSS (Jo et al., 2022)	95.72	98.46	0.0033	97.01	99.64	0.0195
GDSS + TF (Jo et al., 2022)	99.68	-	0.0024	96.04	-	0.0326
GruM (Jo et al., 2024)	99.69	96.90	0.0002	98.65	99.97	0.0015
GraphBFN (Song et al., 2025)	99.73	-	0.0002	99.22	-	0.0013
SID (Boget, 2025)	99.67	95.66	**0.0001**	99.50	99.84	0.0021
DeFoG (Qin et al., 2025)	99.30	96.30	-	99.22	99.99	0.0008
GBD (Liu et al., 2025)	99.88	-	0.0002	97.87	-	0.0018
Flowette (Wijesinghe et al., 2026)	99.81 $\pm$ 0.09	99.30 $\pm$ 0.05	0.0003 $\pm$ 0.0002	99.90 $\pm$ 0.10	**100.00** $\pm$ 0.0	0.0006 $\pm$ 0.0004
GraphonFlow(Fourier)	**99.90** $\pm$ 0.05	**99.40** $\pm$ 0.06	0.0003 $\pm$ 0.0004	99.86 $\pm$ 0.02	**100.00** $\pm$ 0.0	0.0006 $\pm$ 0.0006
GraphonFlow(Bernstein)	**99.93** $\pm$ 0.03	**99.65** $\pm$ 0.02	0.0002 $\pm$ 0.0005	**99.93** $\pm$ 0.04	**100.00** $\pm$ 0.0	**0.0004** $\pm$ 0.0003

Table 2: Generation performance on Guacamol and MOSES molecular graphs.

Model	Guacamol			MOSES			
	Valid ↑	V.U.N ↑	KL div. ↑	Unique ↑	Novelty ↑	SNN ↑	Scaff. ↑
DiGress (Vignac et al., 2023)	85.2	85.1	92.9	**100.0**	95.0	0.52	14.8
DisCo (Xu et al., 2024)	86.6	86.5	92.6	**100.0**	97.7	0.50	15.1
Cometh (Siraudin et al., 2024)	98.9	97.6	96.7	99.9	92.6	0.54	**16.0**
GraphBFN (Song et al., 2025)	-	-	-	99.8	89.0	**0.59**	10.0
G2PT (Chen et al., 2025)	95.3	94.8	95.6	**100.0**	79.4	0.55	2.9
GEEL (Jang et al., 2024)	88.2	77.1	93.1	**100.0**	81.1	0.52	3.6
DeFoG (Qin et al., 2025)	99.0	97.9	97.7	99.9	92.1	0.55	14.4
Flowette (Wijesinghe et al., 2026)	98.6	98.0	**97.9**	99.9	98.10	0.58	15.3
GraphonFlow(Fourier)	**99.05**	**98.50**	97.1	**100.0**	**98.45**	0.57	15.7
GraphonFlow(Bernstein)	**99.30**	**99.00**	97.7	**100.0**	**98.85**	**0.59**	**16.0**

We follow the dataset partitions used in prior graph-generation evaluations (Qin et al., 2025). For Tree and SBM, performance is measured using degree, clustering-coefficient, and orbit-distribution discrepancies together with validity, uniqueness, novelty, and V.U.N.; for Ego-small, we report the three structural distribution metrics. We use the same evaluation metrics for synthetic graph generation as used in Flowette (Wijesinghe et al., 2026). Since these benchmarks do not contain molecular attributes or chemical constraints, the valence regularisation term is disabled by setting $\beta_{\text{val}} = 0$. We evaluate both the Fourier and Bernstein parameterisations of the latent-conditioned graphon. Complete experimental details are provided in Appendix I.1.

Additional Experiments. While the benchmark results evaluate overall generative performance, we further isolate the mechanisms underlying *GraphonFlow* through ablations of the graphon source distribution, canonical coordinate conditioning, Stage-1 and Stage-2 regularisation, and two-stage vs. joint optimisation. We additionally analyse training and sampling runtime under the same computational setting and qualitatively examine the learned molecular representation using LLE visualisations. Complete ablation, runtime, and representation-visualisation results are provided in Appendices I.5, I.3, and I.4, respectively.

6 Conclusion and Limitations

We introduced *GraphonFlow*, a probabilistic graph generation framework that learns a structured source distribution for continuous-time flow matching rather than relying on independent noise or fixed structural priors. *GraphonFlow* combines hybrid graph- and node-level latent representations with a latent-conditioned graphon, canonical graph labelling, connectivity-aware regularisation, and two-stage optimisation to separate structural-prior learning from transport. Across molecular and synthetic benchmarks, *GraphonFlow* achieves competitive performance in validity, diversity, and structural fidelity, while extensive ablations support the contributions of the learned graphon source, canonical coordinates, connectivity-aware objectives, and two-stage training. Current limitations include reliance on deterministic canonicalisation and evaluation on a bounded set of molecular and synthetic graph domains; we discuss these limitations and directions for extension in App. J. Overall, our results demonstrate that learning a latent-conditioned structured source can simplify the transport problem, allowing the flow to refine graph structure rather than construct it entirely from unstructured noise.

Reproducibility Statement

Architectural details are provided in Fig. 1, Appendices G and D, with dataset-specific hyperparameters reported in Appendix I. Dataset evaluation protocols are described in Section 5. Proofs of all theoretical results are given in full in Appendix C. Our code is available at: https://anonymous.4open.science/r/GraphonFlow

Ethics Statement

This work introduces a generative modeling framework for graph generation and relies exclusively on publicly available molecular and synthetic benchmark datasets commonly used in this research area, in accordance with their applicable usage and licensing terms. The study does not involve human participants, sensitive or private information, or the collection of new data. We do not identify any immediate or substantial ethical risks associated with the proposed methodology. The authors have reviewed the study and confirm its adherence to the ICLR Code of Ethics.

AI use statement

Large language models were employed only for refining grammar, manuscript organization, and improving the phrasing of selected passages. The authors independently conceived the research, developed the methodology and code, designed and conducted the experiments, interpreted the results, and prepared the scientific content of the manuscript.

References

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *International Conference on Learning Representations*, volume 2024, pp. 27698–27736, 2024.

Yoann Boget. Simple and critical iterative denoising: A recasting of discrete diffusion in graph generation. In *International Conference on Machine Learning*, 2025.

Christian Borgs, Jennifer T. Chayes, Henry Cohn, and Nina Holden. Sparse exchangeable graphs and their limits via graphon processes. *Journal of Machine Learning Research*, 18(210):1–71, 2018.

Nathan Brown, Marco Fiscato, Marwin H. S. Segler, and Alain C. Vaucher. GuacaMol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3): 1096–1108, 2019.

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

François Caron and Emily B. Fox. Sparse graphs using exchangeable random measures. *Journal of the Royal Statistical Society: Series B*, 79(5):1295–1366, 2017. doi: 10.1111/rssb.12233.

Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning*, pp. 4585–4610. PMLR, 2023.

Xiaohui Chen, Yinkai Wang, Jiaxing He, Yuanqi Du, Soha Hassoun, Xiaolin Xu, and Li-Ping Liu. Graph generative pre-trained transformer. In *International Conference on Machine Learning*, 2025.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International conference on machine learning*, pp. 2302–2312. PMLR, 2020.

Philip J Davis. *Interpolation and approximation*. Courier Corporation, 1975.

Persi Diaconis and Svante Janson. Graph limits and exchangeable random graphs. *arXiv preprint arXiv:0712.2749*, 2007.

Nathaniel Lee Diamant, Alex M Tseng, Kangway V Chuang, Tommaso Biancalani, and Gabriele Scalia. Improving graph generation by restricting graph bandwidth. In *International Conference on Machine Learning*, pp. 7939–7959. PMLR, 2023.

Ian Dunn and David R Koes. Flowmol3: flow matching for 3d de novo small-molecule generation. *Digital Discovery*, 5(5):2052–2066, 2026.

Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *Advances in Neural Information Processing Systems*, 2024.

Mama Foupouagnigni and Merlin Mouafo Wouodjié. On multivariate bernstein polynomials. *Mathematics*, 8(9):1397, 2020.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Daniel Glasscock. What is a graphon? *arXiv preprint arXiv:1611.00718*, 2016.

Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of the web conference 2020*, pp. 1253–1263, 2020.

Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *NeurIPS New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers)*, 2022.

Xiaoyang Hou, Tian Zhu, Milong Ren, Dongbo Bu, Xin Gao, Chunming Zhang, and Shiwei Sun. Improving molecular graph generation with flow matching and optimal transport. *arXiv preprint arXiv:2411.05676*, 2024.

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 (ICDM)*, pp. 201–210. IEEE, 2022.

Yiming Huang and Tolga Birdal. Hog-diff: Higher-order guided diffusion for graph generation. In *International Conference on Learning Representations*, 2026.

Yunhui Jang, Seul Lee, and Sungsoo Ahn. A simple and scalable representation for graph generation. In *International Conference on Learning Representations*, 2024.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pp. 2323–2332. PMLR, 2018.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, 2022.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. In *International Conference on Machine Learning*, 2024.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas N. Kipf and Max Welling. Variational graph auto-encoders. In *NeurIPS Bayesian Deep Learning Workshop*, 2016.

Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation. In *International conference on machine learning*, pp. 17391–17408. PMLR, 2023.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

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

Phillip Lippe and Efstratios Gavves. Categorical normalizing flows via continuous transformations. In *International Conference on Learning Representations*, 2021.

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

Xinyang Liu, Yilin He, Bo Chen, and Mingyuan Zhou. Advancing graph generation through beta diffusion. In *International Conference on Learning Representations*, volume 2025, pp. 82089–82114, 2025.

George G Lorentz. *Bernstein polynomials*. American Mathematical Soc., 2012.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

László Lovász. Large networks and graph limits. *American Mathematical Society*, 60, 2012.

Chunyu Luo, Yuankai Luo, Xiao-Ming Wu, and Lei Shi. Simgfm: Simplifying discrete flow matching for graph generation. In *International Conference on Machine Learning*, 2026.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pp. 15159–15179. PMLR, 2022.

Brendan D McKay. Practical graph isomorphism. 1981.

Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of symbolic computation*, 60:94–112, 2014.

Mark EJ Newman. Random graphs as models of networks. *arXiv preprint cond-mat/0202208*, 2002.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, 2020.

George M. Phillips. *Bernstein Polynomials*, pp. 247–290. Springer New York, New York, NY, 2003. ISBN 978-0-387-21682-9. doi: 10.1007/0-387-21682-0_7. URL https://doi.org/10.1007/0-387-21682-0_7.

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

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. *International Conference on Machine Learning*, 2025.

Erzsebet Ravasz, Anna Lisa Somera, D. A. Mongru, Zoltan N. Oltvai, and Albert-Laszlo Barabasi. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002. doi: 10.1126/science.1073374.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *International Conference on Learning Representations (ICLR)*, 2020.

Antoine Siraudin, Fragkiskos D Malliaros, and Christopher Morris. Cometh: A continuous-time discrete-state graph diffusion model. *Transactions on Machine Learning Research*, 2024.

Yuxuan Song, Juntong Shi, Jingjing Gong, Minkai Xu, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Smooth interpolation for improved discrete graph generative models. In *International Conference on Machine Learning*, 2025.

Teague Sterling and John J. Irwin. ZINC15—ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *International Conference on Learning Representations*, 2023.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998. doi: 10.1038/30918.

Asiri Wijesinghe, Sevvandi Kandanaarachchi, Daniel M Steinberg, and Cheng Soon Ong. Flowette: Flow matching with graphette priors for graph generation. *arXiv preprint arXiv:2602.23566*, 2026.

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

Xinyue Xia, Gal Mishne, and Yusu Wang. Implicit graphon neural representation. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pp. 10619–10634, 2023.

Hongteng Xu, Peilin Zhao, Junzhou Huang, and Dixin Luo. Learning graphon autoencoders for generative graph modeling. *arXiv preprint arXiv:2105.14244*, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. *Advances in Neural Information Processing Systems*, 37:79704–79740, 2024.

Kai Yi, Bingxin Zhou, Yiqing Shen, Pietro Liò, and Yuguang Wang. Graph denoising diffusion for inverse protein folding. *Advances in Neural Information Processing Systems*, 36:10238–10257, 2023.

Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pp. 5708–5717. PMLR, 2018.

Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 617–626, 2020.

Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey on deep graph generation: Methods and applications. In *Learning on Graphs Conference*, pp. 47–1. PMLR, 2022.

A BERNSTEIN GRAPHON BASIS

Recall the basis function

$$B_{k\ell}^{(K)}(x,y) = \frac{1}{2}\left(\phi_{k\ell}^{(K)}(x,y) + \phi_{\ell k}^{(K)}(x,y)\right) \tag{21}$$

where

$$\phi_{k\ell}^{(K)}(x,y) = \binom{K}{k}\binom{K}{\ell} x^k (1-x)^{K-k} y^\ell (1-y)^{K-\ell} \tag{22}$$

for $K \in \mathbb{Z}^+$, $k, \ell \in \{0, \ldots, n\}$, $k \leq \ell$ and $x, y \in [0,1]$.

A.1 PROPERTIES OF THE BERNSTEIN GRAPHON BASIS

For a given $K \in \mathbb{Z}^+$, $k, \ell \in \{0, \ldots, K\}$ and $k \leq \ell$ let $\mathcal{B}^{(K)} = \left\{B_{00}^{(K)} B_{01}^{(K)}, \ldots, B_{KK}^{(K)}\right\}$ denote the graphon basis.

1. All basis functions $B_{k\ell}^{(K)}(x,y)$ satisfy $0 \leq B_{k\ell}^{(K)}(x,y) \leq 1$.
   From Eq. 22 for $x, y \in [0,1]$, $\phi_{k\ell}(x,y) \in [0,1]$ giving rise to $B_{k\ell}^{(K)}(x,y) \in [0,1]$.
2. $B_{k\ell}^{(K)}(x,y) = B_{k\ell}^{(K)}(y,x)$, i.e., the graphon basis functions are symmetric. Note that $\phi_{kl}^{(K)}(y,x) = \phi_{lk}^{(K)}(x,y)$ from Eq. 22. This gives us
$$\begin{aligned} B_{k\ell}^{(K)}(y,x) &= \frac{1}{2}\left(\phi_{k\ell}^{(K)}(y,x) + \phi_{\ell k}^{(K)}(y,x)\right), \\ &= \frac{1}{2}\left(\phi_{\ell k}^{(K)}(x,y) + \phi_{k\ell}^{(K)}(x,y)\right),, \\ &= B_{k\ell}^{(K)}(x,y)\,. \end{aligned}$$
3. $B_{kl}(x,y) = B_{lk}(x,y)$.
   Follows from $B_{k\ell}^{(K)}(x,y) = \frac{1}{2}\left(\phi_{k\ell}^{(K)}(x,y) + \phi_{\ell k}^{(K)}(x,y)\right)$. As we only consider $k \leq \ell$ we only consider one of $B_{k\ell}^{(K)}$ or $B_{\ell k}^{(K)}$.
4. $\sum_{k,l} B_{k\ell}^{(K)}(x,y) = 1$. Let $b_k^{(K)}(x) = \binom{K}{k} x^k (1-x)^{K-k}$. Note that
$$\sum_{k=0}^{K} b_k^{(K)}(x) = \sum_{k=0}^{K} \binom{K}{k} x^k (1-x)^{K-k} = (x+1-x)^K = 1\,,$$
   from the binomial theorem. As $\phi_{k\ell}^{(K)}(x,y) = b_k^{(K)}(x) b_\ell^{(K)}(y)$ we have
$$\sum_{k,l} \phi_{k\ell}^{(K)}(x,y) = \sum_{k=0}^{K}\sum_{\ell=0}^{K} \phi_{k\ell}^{(K)}(x,y) = \sum_{k=0}^{K}\sum_{\ell=0}^{K} b_k^{(K)}(x) b_\ell^{(K)}(y) = \sum_{k=0}^{K} b_k^{(K)}(x) \sum_{\ell=0}^{K} b_\ell^{(K)}(y) = 1\,.$$
   The result follows as $B_{k\ell}^{(K)}(x,y) = \frac{1}{2}\left(\phi_{k\ell}^{(K)}(x,y) + \phi_{\ell k}^{(K)}(x,y)\right)$.
5. The Bernstein basis $\mathcal{B}^{(K)}$ spans polynomial graphons of degree K in x and y because the basis vectors are symmetric polynomials of degree K.

Proposition 1. *Suppose* $W(x,y) \in C\left([0,1]^2\right)$ *and let*

$$\tilde{W}_K(x,y) = \sum_{r=0}^{K}\sum_{s=0}^{K} W\left(\frac{r}{K}, \frac{s}{K}\right)\phi_{rs}^{(K)}(x,y) \quad \textit{and} \quad \widehat{W}_K(x,y) = \frac{1}{2}\left(\tilde{W}_K(x,y) + \tilde{W}_K(y,x)\right)\,,$$

where $\phi_{rs}^{(K)}$ *is defined as in Eq. 22. Then*

$$\lim_{K\to\infty} \widehat{W}_K(x,y) = W(x,y)\,.$$

Proof. The one-dimensional Bernstein convergence for continuous functions is given in Lorentz (2012) and Phillips (2003) Theorem 7.1.5. The multivariate Bernstein convergence is stated in Davis (1975) and later proved in detail in Foupouagnigni & Mouafo Wouodjié (2020), which gives

$$\lim_{K\to\infty} \tilde{W}_K(x,y) = W(x,y)\,. \tag{23}$$

We compute

$$\begin{aligned}\|W(x,y) - \widehat{W}_K(x,y)\| &= \frac{1}{2}\|W(x,y) - \tilde{W}_K(x,y) + W(x,y) - \tilde{W}_K(y,x)\|\,,\\ &\leq \frac{1}{2}\left(\|W(x,y) - \tilde{W}_K(x,y)\| + \|W(x,y) - \tilde{W}_K(y,x)\|\right)\,,\\ &= \|W(x,y) - \tilde{W}_K(x,y)\|\end{aligned}$$

as W is symmetric $\|W(x,y) - \tilde{W}_K(y,x)\| = \|W(y,x) - \tilde{W}_K(y,x)\| = \|W(x,y) - \tilde{W}_K(x,y)\|$. Thus we get the result from Eq. 23.

□

A.2 An example basis

Consider the graphon basis for $K = 2$. Then $\mathcal{B} = \{B_{00}, B_{01}, B_{02}, B_{11}, B_{12}, B_{22}\}$. This basis is shown in Figure 4.

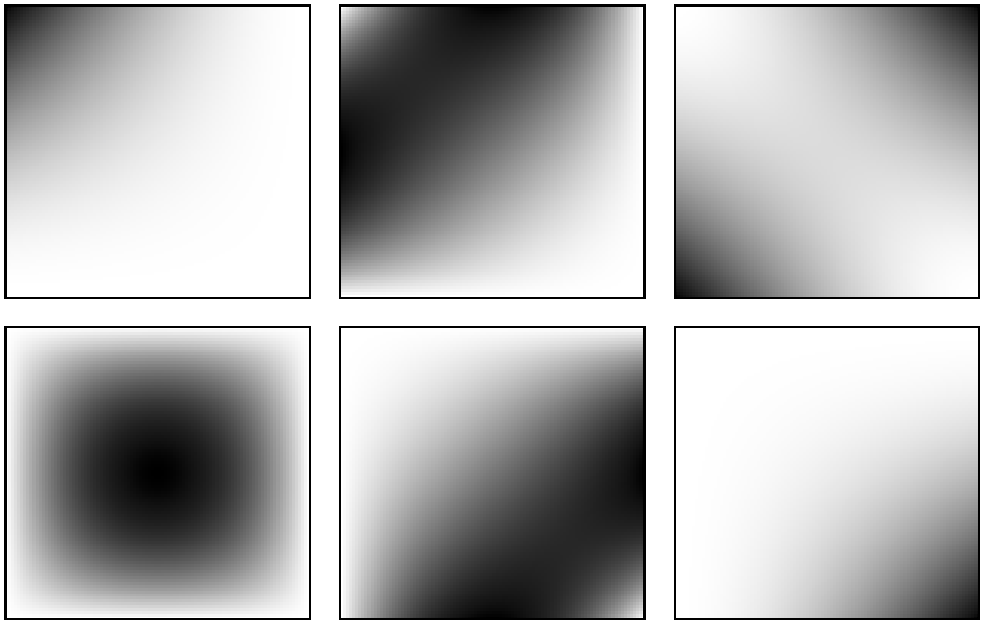


Figure 4: Bernstein graphon basis for $K = 2$

Let us suppose we want to estimate the graphon $W = \exp(-2(x+y))$. We get the following estimate

$$\hat{W} = 0.96B^{(2)}_{00} + 0.35B^{(2)}_{01} + 0.13B^{(2)}_{02} + 0.13B^{(2)}_{11} + 0.05B^{(2)}_{12} + 0.02B^{(2)}_{22}\,. \tag{24}$$

Figure 5 shows the two graphons W and $\hat{W}$.

B A symmetric Fourier basis

Let M denote the maximum frequency of the Fourier basis, i.e., we consider cosine and sine terms in the form of $\cos(2\pi mx)$ or $\sin(2\pi mx)$ for $m \in \{1,\ldots,M\}$. We consider the following symmetric, basis functions on $[0,1]^2$.

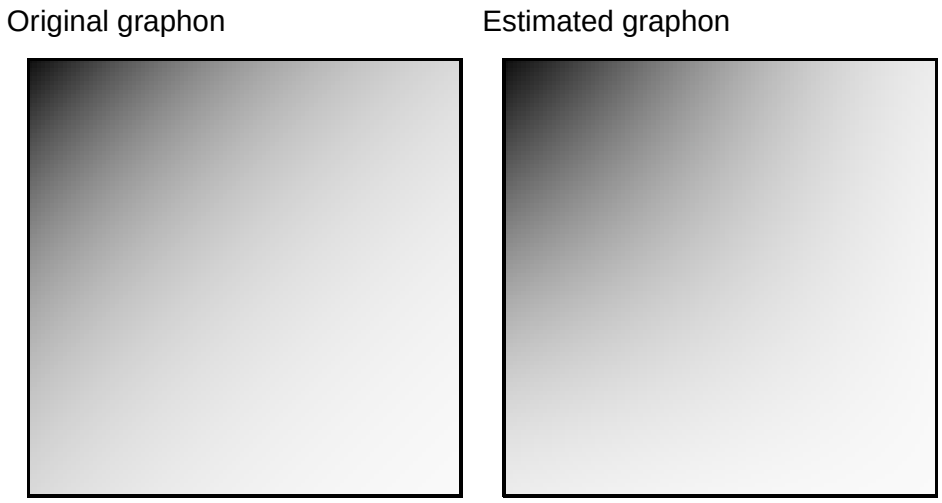


Figure 5: Graphons $W = \exp(-2(x+y))$ on the left and estimated graphon $\hat{W}$ in Eq. 24 on the right.

1. The constant basis function is $f_{00}(x, y) = 1$.
2. Two sets of non-product basis functions are given by
   (a) $f_{0c_m}(x, y) = \frac{1}{2}(\cos(2\pi mx) + \cos(2\pi my))$ and
   (b) $f_{0s_m}(x, y) = \frac{1}{2}(\sin(2\pi mx) + \sin(2\pi my))$
   for $m \in \{1, \ldots, M\}$.
3. Four sets of product basis functions are given by
   (a) $f_{c_m c_n} = \frac{1}{2}(\cos(2\pi mx)\cos(2\pi ny) + \cos(2\pi nx)\cos(2\pi my))$,
   (b) $f_{s_m s_n} = \frac{1}{2}(\sin(2\pi mx)\sin(2\pi ny) + \sin(2\pi nx)\sin(2\pi my))$,
   (c) $f_{c_m s_n} = \frac{1}{2}(\cos(2\pi mx)\sin(2\pi ny) + \sin(2\pi nx)\cos(2\pi my))$ and
   (d) $f_{c_n s_m} = \frac{1}{2}(\cos(2\pi nx)\sin(2\pi my) + \sin(2\pi mx)\cos(2\pi ny))$
   for $m, n \in \{1, \ldots, M\}$ and $m \leq n$, with the caveat that when $m = n$, $f_{c_m s_n} = f_{c_n s_m}$. Thus, for $m = n$ only one basis function of $f_{c_m s_n}$ and $f_{c_m s_n}$ needs to be considered.

Alternatively, instead of 3 above, we can consider two separate basis function sets as follows:

3(i). (a) $f_{c_m c_n} = \frac{1}{2}(\cos(2\pi mx)\cos(2\pi ny) + \cos(2\pi nx)\cos(2\pi my))$,
(b) $f_{s_m s_n} = \frac{1}{2}(\sin(2\pi mx)\sin(2\pi ny) + \sin(2\pi nx)\sin(2\pi my))$,
(c) $f_{c_m s_n} = \frac{1}{2}(\cos(2\pi mx)\sin(2\pi ny) + \sin(2\pi nx)\cos(2\pi my))$ and
(d) $f_{c_n s_m} = \frac{1}{2}(\cos(2\pi nx)\sin(2\pi my) + \sin(2\pi mx)\cos(2\pi ny))$
for $m, n \in \{1, \ldots, M\}$ and $m < n$, and

3(ii). (a) $f_{c_m c_m} = \cos(2\pi mx)\cos(2\pi my)$,
(b) $f_{s_m s_m} = \sin(2\pi mx)\sin(2\pi my)$ and
(c) $f_{c_m s_m} = \frac{1}{2}(\cos(2\pi mx)\sin(2\pi my) + \sin(2\pi mx)\cos(2\pi my))$.

For $M = 2$ we get the the basis $\mathcal{B} = \{f_{00}, f_{0c_1}, f_{0s_1}, f_{0c_2}, f_{0s_2}, f_{c_1c_1}, f_{s_1s_1}, f_{c_1s_1}, f_{c_2c_2}, f_{s_2s_2}, f_{c_2s_2}, f_{c_1c_2}, f_{s_1s_2}, f_{c_1s_2}, f_{c_2s_1}\}$.

Thus a linear combination of the basis functions is given by

$$g(x, y) = \sum_{i \in K} a_k f_k$$

where k describes the set of indices in the basis vectors. In the above example with $M = 2$, the set $K = \{00, 0c_1, 0s_1, 0c_2, 0s_2, c_1c_1, s_1s_1, c_1s_1, c_2c_2, s_2s_2, c_2s_2, c_1c_2, s_1s_2, c_1s_2, c_2s_1\}$

Note that the symmetric Fourier basis is **not** a graphon basis. Each function f_k, defined on $[0,1]^2$ has the range $[-1,1]$. Hence a linear combination $\sum_{i\in K} a_k f_k \in \mathbb{R}$, not restricted to $[0,1]$. Notwithstanding this limitation, a graphon can be approximated by a linear combination of the symmetric Fourier basis.

Let $W(x,y)$ be a graphon and $\hat{W}(x,y)$ be its approximation such that

$$\hat{W}(x,y) = \sum_{i\in K} a_k f_k \,.$$

Then the coefficients a_k can be obtained by computing a Fast Discrete Fourier Transform.

B.1 Fourier coefficients

Let a function $f : [0,1]^2 \to \mathbb{R}$ be sampled on an $N \times N$ grid. Suppose

$$f_{k\ell} = f\left(\frac{k}{N}, \frac{\ell}{N}\right)$$

for $k, \ell \in \{0, \ldots, N-1\}$. The standard Fourier basis functions are $\exp(2\pi i(mx+ny))$ for $m, n \in \mathbb{Z}$.

We can write f as a linear combination of the basis functions as follows

$$f(x,y) = \sum_{m,n\in\mathbb{Z}} \hat{f}(m,n) e^{2\pi i(mx+ny)} \,. \tag{25}$$

The Fourier coefficients via the Fourier transform are given by

$$\hat{f}(m,n) = \int_0^1 \int_0^1 f(x,y) e^{-2\pi i(mx+ny)}\, dxdy \,.$$

The 2D Discrete Fourier Transformation (DFT) is an approximation of the continuous Fourier transform, and thus gives

$$\hat{f}(m,n) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} f_{k\ell} e^{-2\pi i\left(\frac{mk}{N} + \frac{n\ell}{N}\right)} \,. \tag{26}$$

Expanding Eq. 25 we get

$$\begin{aligned}
f(x,y) &= \sum_{m,n\in\mathbb{Z}} \hat{f}(m,n) e^{2\pi i(mx+ny)} \,, \\
&= \hat{f}(0,0) + \sum_{m=0,n\in\mathbb{Z}} \hat{f}(0,n) e^{2\pi i(ny)} + \sum_{m\in\mathbb{Z},n=0} \hat{f}(m,0) e^{2\pi i(mx)} + \sum_{m,n\in\mathbb{Z}^+} \Big(\hat{f}(m,n) e^{2\pi i(mx+ny)} && (27) \\
&+ \hat{f}(-m,n) e^{2\pi i(-mx+ny)} + \hat{f}(m,-n) e^{2\pi i(mx-ny)} + \hat{f}(-m,-n) e^{-2\pi i(mx+ny)} \Big) && (28) \\
&= \hat{f}(0,0) + \sum_{m=0,n\in\mathbb{Z}^+} \Big(\hat{f}(0,n) e^{2\pi iny} + \hat{f}(0,-n) e^{-2\pi iny} \Big) && (29) \\
&+ \sum_{m\in\mathbb{Z}^+,n=0} \Big(\hat{f}(m,0) e^{2\pi imx} + \hat{f}(-m,0) e^{-2\pi imx} \Big) + \sum_{m,n\in\mathbb{Z}^+} \Big(\hat{f}(m,n) e^{2\pi i(mx+ny)} && (30) \\
&+ \hat{f}(-m,n) e^{2\pi i(-mx+ny)} + \hat{f}(m,-n) e^{2\pi i(mx-ny)} + \hat{f}(-m,-n) e^{-2\pi i(mx+ny)} \Big) && (31)
\end{aligned}$$

Since f is real, we have conjugate symmetry giving us

$$\hat{f}(m,-n) = \overline{\hat{f}(-m,n)} \,.$$

Thus we get

$$
\begin{aligned}
f(x,y) = \hat{f}(0,0) &+ \sum_{m=0,n\in\mathbb{Z}^+} \left(\hat{f}(0,n)e^{2\pi iny} + \overline{\hat{f}(0,n)}e^{-2\pi iny}\right) \\
&+ \sum_{m\in\mathbb{Z}^+,n=0} \left(\hat{f}(m,0)e^{2\pi imx} + \overline{\hat{f}(m,0)}e^{-2\pi imx}\right) \\
&+ \sum_{m,n\in\mathbb{Z}^+} \left(\hat{f}(m,n)e^{2\pi i(mx+ny)} + \overline{\hat{f}(m,n)}e^{-2\pi i(mx+ny)}\right) \\
&+ \sum_{m,n\in\mathbb{Z}^+} \left(\hat{f}(m,-n)e^{2\pi i(mx-ny)} + \overline{\hat{f}(m,-n)}e^{-2\pi i(mx-ny)}\right) \qquad (32)
\end{aligned}
$$

Let us consider the generic expression $\hat{f}(a,b)e^{i\theta} + \overline{\hat{f}(a,b)}e^{-i\theta}$ which appears multiple times in Eq. 32. Let $\hat{f}(a,b) = p + iq$ Then we get

$$
\begin{aligned}
\hat{f}(a,b)e^{i\theta} + \overline{\hat{f}(a,b)}e^{-i\theta} &= (p+iq)(\cos\theta + i\sin\theta) + (p-iq)(\cos\theta - i\sin\theta)\,, \\
&= 2(p\cos\theta - q\sin\theta)\,, \qquad (33) \\
&= 2\Re\left(\hat{f}(a,b)e^{i\theta}\right)\,, \qquad (34)
\end{aligned}
$$

where $\Re$ denotes the real part of a complex number. Thus, Eq. 32 can be written as

$$
\begin{aligned}
f(x,y) = \hat{f}(0,0) &+ \sum_{m=0,n\in\mathbb{Z}^+} 2\Re\left(\hat{f}(0,n)e^{2\pi iny}\right) \\
&+ \sum_{m\in\mathbb{Z}^+,n=0} 2\Re\left(\hat{f}(m,0)e^{2\pi imx}\right) \\
&+ \sum_{m,n\in\mathbb{Z}^+} 2\Re\left(\hat{f}(m,n)e^{2\pi i(mx+ny)}\right) \\
&+ \sum_{m,n\in\mathbb{Z}^+} 2\Re\left(\hat{f}(m,-n)e^{2\pi i(mx-ny)}\right) \qquad (35)
\end{aligned}
$$

Let $\hat{f}(m,n) = P_{mn} + iQ_{mn}$ and $\hat{f}(m,-n) = P_{mn_-} + iQ_{mn_-}$. Using Eq. 33 we have

$$
\begin{aligned}
\Re\left(\hat{f}(0,n)e^{2\pi iny}\right) &= P_{0n}\cos(2\pi ny) - Q_{0n}\sin(2\pi ny)\,, \qquad (36) \\
\Re\left(\hat{f}(m,0)e^{2\pi imx}\right) &= P_{m0}\cos(2\pi mx) - Q_{m0}\sin(2\pi mx)\,, \qquad (37) \\
\Re\left(\hat{f}(m,n)e^{2\pi i(mx+ny)}\right) &= P_{mn}\cos 2\pi(mx+ny) - Q_{mn}\sin 2\pi(mx+ny)\,, \qquad (38) \\
&= P_{mn}\cos(2\pi mx)\cos(2\pi ny) - P_{mn}\sin(2\pi mx)\sin(2\pi ny) \qquad (39) \\
&\quad - Q_{mn}\sin(2\pi mx)\cos(2\pi ny) - Q_{mn}\cos(2\pi mx)\sin(2\pi ny)\,, \qquad (40) \\
\Re\left(\hat{f}(m,-n)e^{2\pi i(mx-ny)}\right) &= P_{mn_-}\cos 2\pi(mx-ny) - Q_{mn_-}\sin 2\pi(mx-ny) \qquad (41) \\
&= P_{mn_-}\cos(2\pi mx)\cos(2\pi ny) + P_{mn_-}\sin(2\pi mx)\sin(2\pi ny) \qquad (42) \\
&\quad - Q_{mn_-}\sin(2\pi mx)\cos(2\pi ny) + Q_{mn_-}\cos(2\pi mx)\sin(2\pi ny)\,, \qquad (43)
\end{aligned}
$$

Substituting in Eq. 35 we get

$$\begin{aligned} f(x,y) = \hat{f}(0,0) &+ \sum_{m=0,n\in\mathbb{Z}^+} 2P_{0n}\cos(2\pi ny) + (-2Q_{0n})\sin(2\pi ny) \\ &+ \sum_{m\in\mathbb{Z}^+,n=0} 2P_{m0}\cos(2\pi mx) + (-2Q_{m0})\sin(2\pi mx) \\ &+ \sum_{m,n\in\mathbb{Z}^+} 2P_{mn}\cos(2\pi mx)\cos(2\pi ny) + (-2P_{mn})\sin(2\pi mx)\sin(2\pi ny) \\ &+ (-2Q_{mn})\sin(2\pi mx)\cos(2\pi ny) + (-2Q_{mn})\cos(2\pi mx)\sin(2\pi ny)\,, \\ &+ \sum_{m,n\in\mathbb{Z}^+} 2P_{mn_}\cos(2\pi mx)\cos(2\pi ny) + 2P_{mn_}\sin(2\pi mx)\sin(2\pi ny) \\ &+ (-2Q_{mn_})\sin(2\pi mx)\cos(2\pi ny) + 2Q_{mn_}\cos(2\pi mx)\sin(2\pi ny)\,. \end{aligned} \tag{44}$$

Grouping the terms in Eq. 44 we get

$$\begin{aligned} f(x,y) = \hat{f}(0,0) &+ \sum_{m=0,n\in\mathbb{Z}^+} 2P_{0n}\cos(2\pi ny) + (-2Q_{0n})\sin(2\pi ny) \\ &+ \sum_{m\in\mathbb{Z}^+,n=0} 2P_{m0}\cos(2\pi mx) + (-2Q_{m0})\sin(2\pi mx) \\ &+ \sum_{m,n\in\mathbb{Z}^+} (2P_{mn} + 2P_{mn_})\cos(2\pi mx)\cos(2\pi ny) \\ &+ \sum_{m,n\in\mathbb{Z}^+} ((-2Q_{mn}) + (-2Q_{mn_}))\sin(2\pi mx)\cos(2\pi ny) \\ &+ \sum_{m,n\in\mathbb{Z}^+} ((-2Q_{mn}) + (2Q_{mn_}))\cos(2\pi mx)\sin(2\pi ny) \\ &+ \sum_{m,n\in\mathbb{Z}^+} ((-2P_{mn}) + 2P_{mn_})\sin(2\pi mx)\sin(2\pi ny) \end{aligned} \tag{45}$$

Hence we get the following coefficients for the trigonometric terms:

$$\text{Coef}(1) = \hat{f}(0,0) = \Re\left(\hat{f}(0,0)\right)\,, \tag{46}$$

$$\text{Coef}(\cos(2\pi ny)) = 2P_{0n} = 2\Re\left(\hat{f}(0,n)\right)\,, \tag{47}$$

$$\text{Coef}(\sin(2\pi ny)) = (-2Q_{0n}) = -2\Im\left(\hat{f}(0,n)\right)\,, \tag{48}$$

$$\text{Coef}(\cos(2\pi mx)) = 2P_{m0} = 2\Re\left(\hat{f}(m,0)\right)\,, \tag{49}$$

$$\text{Coef}(\sin(2\pi mx)) = -2Q_{m0} = -2\Im\left(\hat{f}(m,0)\right)\,, \tag{50}$$

$$\text{Coef}\left(\cos(2\pi mx)\cos(2\pi ny)\right) = (2P_{mn} + 2P_{mn_}) = 2\Re\left(\hat{f}(m,n) + \hat{f}(m,-n)\right)\,, \tag{51}$$

$$\text{Coef}\left(\sin(2\pi mx)\cos(2\pi ny)\right) = (-2Q_{mn}) + (-2Q_{mn_}) = -2\Im\left(\hat{f}(m,n) + \hat{f}(m,-n)\right)\,, \tag{52}$$

$$\text{Coef}\left(\cos(2\pi mx)\sin(2\pi ny)\right) = ((-2Q_{mn}) + (2Q_{mn_})) = -2\Im\left(\hat{f}(m,n) - \hat{f}(m,-n)\right)\,, \tag{53}$$

$$\text{Coef}\left(\sin(2\pi mx)\sin(2\pi ny)\right) = ((-2P_{mn}) + 2P_{mn_}) = -2\Re\left(\hat{f}(m,n) - \hat{f}(m,-n)\right)\,, \tag{54}$$

where $\Im$ denotes the imaginary part of a complex number. The above functions form a Fourier basis, however they are not symmetric in $[0,1]^2$, apart from the constant function.

The standard Fourier coefficients of f are given by Eq. 44. Now suppose f is symmetric, i.e. $f(x,y) = f(y,x)$. Then we first compute the standard Fourier coefficients as in Eq. 44 and

symmetrize it as follows:

$$\hat{f}_{\text{sym}}(m,n) = \frac{1}{2}\left(\hat{f}(m,n) + \hat{f}(n,m)\right) , \tag{55}$$

where $\hat{f}_{\text{sym}}$ denotes the symmetric Fourier coefficients. We symmetrize the Fourier coefficients in Eq. 44 as below:

$$\begin{aligned} f(x,y) = \hat{f}(0,0) + \sum_{m\in\mathbb{Z}^+} & \left[A_{m0}\left(\cos(2\pi mx) + \cos(2\pi my)\right)\right. \\ & \left. + B_{m0}\left(\sin(2\pi mx) + \sin(2\pi my)\right)\right] \\ + \sum_{\substack{m,n\in\mathbb{Z}^+ \\ m>n}} & \left[A_{mn}\left(\cos(2\pi mx)\cos(2\pi ny) + \cos(2\pi nx)\cos(2\pi my)\right)\right. \\ & + B_{mn}\left(\sin(2\pi mx)\cos(2\pi ny) + \cos(2\pi nx)\sin(2\pi my)\right) \\ & + C_{mn}\left(\cos(2\pi mx)\sin(2\pi ny) + \sin(2\pi nx)\cos(2\pi my)\right) \\ & \left. + D_{mn}\left(\sin(2\pi mx)\sin(2\pi ny) + \sin(2\pi nx)\sin(2\pi my)\right)\right] . \end{aligned} \tag{56}$$

From Eq. 45 and Eq. 56 we obtain

$$A_{m0} = P_{0m} + P_{m0} = \Re\left(\hat{f}(0,m) + \hat{f}(m,0)\right) , \tag{57}$$

$$B_{m0} = -Q_{0m} - Q_{m0} = -\Im\left(\hat{f}(m,0) + \hat{f}(0,m)\right) , \tag{58}$$

$$A_{mn} = P_{mn} + P_{mn_-} + P_{nm} + P_{nm_-} = \Re\left(\hat{f}(m,n) + \hat{f}(m,-n) + \hat{f}(n,m) + \hat{f}(n,-m)\right) , \tag{59}$$

$$B_{mn} = -Q_{mn} - Q_{mn_-} - Q_{nm} + Q_{nm_-} = -\Im\left(\hat{f}(m,n) + \hat{f}(m,-n) + \hat{f}(n,m) - \hat{f}(n,-m)\right) , \tag{60}$$

$$C_{mn} = -Q_{mn} + Q_{mn_-} - Q_{nm} - Q_{nm_-} = -\Im\left(\hat{f}(m,n) - \hat{f}(m,-n) + \hat{f}(n,m) + \hat{f}(n,-m)\right) , \tag{61}$$

$$D_{mn} = -P_{mn} + P_{mn_-} - P_{nm} + P_{nm_-} = -\Re\left(\hat{f}(m,n) - \hat{f}(m,-n) + \hat{f}(n,m) - \hat{f}(n,-m)\right) . \tag{62}$$

For a function f given in an $N \times N$ matrix, the Discrete Fourier Transform (DFT) will also be given in an $N \times N$ matrix with $\hat{f}(m,-n) = \hat{f}(m,N-n)$ making the above computation easy.

Proposition 2 (Fourier approximation)**.** *For any graphon $W \in L^2([0,1]^2)$, there exist symmetric Fourier coefficients of order M such that $\|W - \hat{W}_M\|_{L^2} \to 0$ as $M \to \infty$.*

Proof. Let the standard Fourier basis vectors of order M be denoted by $\{g_i\}_i$. Then we construct the symmetric basis vectors f_i as

$$f_i(x,y) = \frac{1}{2}\left(g_i(x,y) + g_i(y,x)\right) . \tag{63}$$

Let $\widetilde{W}_M(x,y)$ be the approximation from the standard Fourier transform, i.e.,

$$\widetilde{W}_M(x,y) = \sum_i c_i g_i(x,y) .$$

Let $\hat{W}_M$ be the approximation from the symmetric Fourier transform. We obtain $\hat{W}_M$

$$\begin{aligned} \hat{W}_M &= \frac{1}{2}\left(\widetilde{W}_M(x,y) + \widetilde{W}_M(y,x)\right) , \\ &= \frac{1}{2}\left(\sum_i c_i g_i(x,y) + \sum_i c_i g_i(y,x)\right) , \\ &= \sum_i c_i f_i(x,y) , \end{aligned}$$

where we have used Eq. 55 and Eq. 63. Then

$$\begin{aligned}\|W - \hat{W}_M\|_{L^2} &= \frac{1}{2}\|W(x,y) - \widetilde{W}_M(x,y) + W(x,y) - \widetilde{W}(y,x)\|_{L^2}\,,\\ &\leq \frac{1}{2}\|W(x,y) - \widetilde{W}_M(x,y)\|_{L^2} + \frac{1}{2}\|W(x,y) - \widetilde{W}_M(y,x)\|_{L^2}\,,\\ &= \frac{1}{2}\|W(x,y) - \widetilde{W}_M(x,y)\|_{L^2} + \frac{1}{2}\|(W(x,y) - \widetilde{W}_M(x,y))^T\|_{L^2}\,,\\ &= \|W(x,y) - \widetilde{W}_M(x,y)\|_{L^2}\end{aligned}$$

For the standard Fourier approximation

$$\lim_{M\to\infty}\|W(x,y) - \widetilde{W}_M(x,y)\|_{L^2} = 0\,,$$

making

$$\lim_{M\to\infty}\|W(x,y) - \hat{W}_M(x,y)\|_{L^2} = 0\,.$$

□

C Proofs for Latent-Conditioned Graphon Framework for Structure-Preserving Flow Matching (Section 3)

Theorem 3 (Permutation invariance of canonical graphon supervision)**.** *By the canonicalisation property introduced in Section 2.1, $G \cong G' \Leftrightarrow \mathcal{C}(G) = \mathcal{C}(G')$, for every permutation matrix P and every z_g,*

$$\mathcal{L}_{\text{graphon}}(P\cdot G, z_g) = \mathcal{L}_{\text{graphon}}(G, z_g).$$

Assume further that the node encoder is permutation equivariant and the graph-level posterior is obtained from the permutation-invariant mean/sum readout of Eq. 91. Then $q_\phi(z_g \mid P\cdot G) = q_\phi(z_g \mid G)$ for every permutation P. Then consequently

$$\mathbb{E}_{q_\phi(z_g|P\cdot G)}\big[\mathcal{L}_{\text{graphon}}(P\cdot G, z_g)\big] = \mathbb{E}_{q_\phi(z_g|G)}\big[\mathcal{L}_{\text{graphon}}(G, z_g)\big].$$

If the canonicalisation map $\mathcal{C}$ is independent of the graphon parameters ψ, the same equality holds for the gradients with respect to the graphon parameters:

$$\nabla_\psi\,\mathbb{E}\big[\mathcal{L}_{\text{graphon}}(P\cdot G, z_g)\big] = \nabla_\psi\,\mathbb{E}\big[\mathcal{L}_{\text{graphon}}(G, z_g)\big].$$

Proof. Fix a graph G with n nodes, a permutation matrix $P \in \{0,1\}^{n\times n}$, and a latent variable z_g. We prove the three claims in turn.

Step 1: The canonical lattice is invariant under P. The graphon is evaluated at the deterministic canonical coordinates $s_i = (i-1)/(n-1)$, $i = 1,\dots,n$. This lattice depends only on the number of nodes n, not on the vertex labelling of G. Since permuting the vertices of a graph does not change its node count, $n(P\cdot G) = n(G) = n$, and therefore the coordinate sequence $s_{1:n}$ used to evaluate $W_\psi(\cdot,\cdot \mid z_g)$ is identical for G and for $P\cdot G$. Consequently, unlike the situation in Theorem 9 where $s_{1:n}$ is an explicit conditioning argument of a distribution over latent variables, here $s_{1:n}$ never needs to be written as a separate argument of $\mathcal{L}_{\text{graphon}}$: it is fully determined by n, and n is itself invariant under relabelling.

Step 2: The canonical adjacency target is invariant under P. By definition, G and $P\cdot G$ are isomorphic graphs that differ only by a relabelling of their vertices, i.e. $G \cong P\cdot G$. By the defining property of the canonicalisation map $\mathcal{C}$, $G \cong G' \iff \mathcal{C}(G) = \mathcal{C}(G')$, so in particular $\mathcal{C}(P\cdot G) = \mathcal{C}(G)$. Writing $\widehat{G} = \mathcal{C}(G) = (\widehat{A}, \widehat{X}, \widehat{E})$ for the shared canonical representative, the canonical adjacency matrix $\widehat{A}$ used in the graphon reconstruction loss (Eq. 11) is therefore identical whether it is computed from G or from $P\cdot G$.

Step 3: Pointwise equality of the loss. By Eq. 11, the canonical graphon loss is

$$\mathcal{L}_{\text{graphon}}(G, z_g) = -\frac{1}{\binom{n}{2}}\sum_{1\leq i<j\leq n}\Big[\widehat{A}_{ij}\,\log W_\psi(s_i, s_j \mid z_g) + (1-\widehat{A}_{ij})\,\log\big(1 - W_\psi(s_i, s_j \mid z_g)\big)\Big],$$

a deterministic function of the triple $\big(\widehat{A},\ s_{1:n},\ z_g\big)$ only - it does not depend on G's original vertex labelling in any other way. By Step 1, $s_{1:n}$ is the same for G and $P \cdot G$; by Step 2, $\widehat{A}$ is the same for G and $P \cdot G$; and z_g is held fixed by hypothesis. Since all three arguments coincide,

$$\mathcal{L}_{\text{graphon}}(P \cdot G, z_g) = \mathcal{L}_{\text{graphon}}(G, z_g) \qquad \text{for every } P \text{ and every } z_g,$$

which proves the first claim. Note that this equality holds pointwise, for every fixed value of z_g, and does not yet require any assumption on the encoder q_ϕ.

Step 4: Equality of expectations under the encoder posterior. By hypothesis, the graph-level encoder is permutation invariant: $q_\phi(z_g \mid P \cdot G) = q_\phi(z_g \mid G) \qquad$ for every permutation P, which holds because the graph-level posterior parameters are read out via the mean/sum-pooling of Eq. 91,

$$h_G = \big[\, h_G^{\text{mean}} \,\|\, h_G^{\text{sum}} \,\big], \qquad h_G^{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} h_i, \qquad h_G^{\text{sum}} = \sum_{i=1}^{n} h_i,$$

both of which are symmetric functions of the (multi)set of node embeddings $\{h_i\}_{i=1}^n$ and are therefore unchanged under any reordering of the summation index induced by P. Consequently $q_\phi(\cdot \mid P \cdot G)$ and $q_\phi(\cdot \mid G)$ are the *same distribution* over z_g. Combined with the pointwise function equality of Step 3, integrating an identical function against an identical distribution yields

$$\mathbb{E}_{q_\phi(z_g \mid P\cdot G)}\big[\mathcal{L}_{\text{graphon}}(P\cdot G, z_g)\big] = \mathbb{E}_{q_\phi(z_g \mid G)}\big[\mathcal{L}_{\text{graphon}}(P\cdot G, z_g)\big] = \mathbb{E}_{q_\phi(z_g \mid G)}\big[\mathcal{L}_{\text{graphon}}(G, z_g)\big],$$

where the first equality uses $q_\phi(z_g \mid P \cdot G) = q_\phi(z_g \mid G)$ and the second uses the pointwise equality of Step 3 inside the (now shared) expectation. This proves the second claim.

Step 5: Equality of gradients with respect to ψ. Suppose further that the canonicalisation map $\mathcal{C}$ does not depend on the graphon parameters ψ (as is the case in our implementation, where canonicalisation is computed via McKay's NAUTY algorithm applied to (A, X) alone, with no gradient permitted to flow through this step). Then the equality of Step 4,

$$\mathbb{E}_{q_\phi(z_g \mid P\cdot G)}\big[\mathcal{L}_{\text{graphon}}(P \cdot G, z_g)\big] = \mathbb{E}_{q_\phi(z_g \mid G)}\big[\mathcal{L}_{\text{graphon}}(G, z_g)\big],$$

holds as an identity between two functions of ψ for *every* value of ψ, since $\widehat{A}$ (via $\mathcal{C}$), $s_{1:n}$, and q_ϕ are all independent of ψ, and the only ψ-dependence on either side enters through $W_\psi(\cdot, \cdot \mid z_g)$ inside $\mathcal{L}_{\text{graphon}}$ in exactly the same functional form on both sides. Differentiating both sides of this identity with respect to ψ therefore gives

$$\nabla_\psi\, \mathbb{E}\big[\mathcal{L}_{\text{graphon}}(P \cdot G, z_g)\big] = \nabla_\psi\, \mathbb{E}\big[\mathcal{L}_{\text{graphon}}(G, z_g)\big],$$

which proves the third claim and completes the proof. □

Proposition 4 (Permutation invariance with canonical-position conditioning)**.** *Let C be a deterministic canonicalisation map satisfying $C(P\cdot G) = C(G)$ for every node permutation P. Let $s_{1:n}$ denote the deterministic coordinate sequence associated with the canonical representation $C(G)$. Then conditioning the node prior on canonical position, $p_\omega(z_{1:n} \mid z_g, s_{1:n}) = \prod_{i=1}^n p_\omega(z_i \mid z_g, s_i)$, does not introduce dependence on the arbitrary observed node ordering. In particular, the induced likelihood on unlabeled graphs, $p_\Theta(G) := p_\Theta(C(G))$, satisfies $p_\Theta(P\cdot G) = p_\Theta(G)$ for every permutation P.*

Proof. Fix a graph G with n nodes and a permutation matrix P. **Step 1: The canonical graph and its lattice are invariant under P.** Since G and $P \cdot G$ are isomorphic, the canonicalisation property gives $C(P \cdot G) = C(G) =: \widehat{G}$. Moreover the canonical lattice $s_{1:n}$ used in Eq. 8 depends only on the node count n, and $n(P \cdot G) = n(G) = n$ since relabelling vertices does not change how many there are. Hence $\widehat{G}$ and $s_{1:n}$ are identical whether computed from G or from $P \cdot G$.

Step 2: The generative marginal is a function of $(\widehat{G}, s_{1:n})$ only. By the generative factorisation of Eq. 5, the marginal likelihood assigned by the model to a canonical graph $\widehat{G}$ is

$$p_\Theta(\widehat{G} \mid s_{1:n}) = \iint p(z_g)\, p_\omega(z_{1:n} \mid z_g, s_{1:n})\, p_\Theta(\widehat{G} \mid z_g, z_{1:n}, s_{1:n})\, dz_g\, dz_{1:n}.$$

The integration variables z_g and $z_{1:n}$ are dummy variables of the generative model, drawn according to the fixed prior $p(z_g) = \mathcal{N}(0, I)$ and the position-conditioned prior $p_\omega(z_{1:n} \mid z_g, s_{1:n})$ respectively;

neither depends on any particular vertex labelling of the graph being scored, and no encoder q_ϕ appears anywhere in this expression. Consequently the value of this integral is entirely determined by the pair $(\widehat{G}, s_{1:n})$ that is plugged into the integrand, and by nothing else.

Step 3: Conclude. By Step 1, $C(P \cdot G) = C(G) = \widehat{G}$ and $s_{1:n}(P \cdot G) = s_{1:n}(G)$. Since, by Step 2, $p_\Theta(\,\cdot \mid s_{1:n})$ is a well-defined function of $(\widehat{G}, s_{1:n})$ alone, evaluating it at the identical pair $(\widehat{G}, s_{1:n})$ obtained from either G or $P \cdot G$ yields the identical value:

$$p_\Theta\big(C(P \cdot G)\big) = p_\Theta\big(\widehat{G} \mid s_{1:n}\big) = p_\Theta\big(C(G)\big).$$

By definition $p_\Theta(G) := p_\Theta(C(G))$, so this proves $p_\Theta(P \cdot G) = p_\Theta(G)$ for every permutation P. □

Theorem 5 (Graphon density bound and Erdős–Rényi connectivity comparison)**.** *If $\mathcal{L}_{\text{conn}}(z_g, n) = 0$, then*

$$\bar{W}_n(z_g) \;\geq\; \delta, \tag{64}$$

where $\bar{W}_n(z_g) = \bar{W}_\psi(z_g)\,\rho(z_g)$ is the effective mean edge probability. Since the unsparsified graphon values satisfy $W_\psi(s_i, s_j \mid z_g) \in (0,1)$, their mean also satisfies $\bar{W}_\psi(z_g) \in (0,1)$. Therefore, using $\bar{W}_n(z_g) = \bar{W}_\psi(z_g)\rho(z_g)$ together with Eq. 64, we obtain

$$\rho(z_g) = \frac{\bar{W}_n(z_g)}{\bar{W}_\psi(z_g)} \;\geq\; \bar{W}_n(z_g) \;\geq\; \delta. \tag{65}$$

Unlike the classical giant-component threshold $1/n$, this bound is a fixed constant, independent of n. Consequently, under the corresponding homogeneous Erdős–Rényi comparison with the same mean edge probability, for n large enough that $\ln n/n < \delta$, a graph drawn from $\mathbb{G}(n, \bar{W}_n(z_g))$ would be connected*–not merely contain a giant component–with probability approaching 1 as $n \to \infty$, by the classical sharp connectivity threshold for Erdős–Rényi graphs (Newman, 2002). For smaller n (e.g. QM9, where $n \leq 9$), Eq. 64 still guarantees the weaker density floor $\bar{W}_n(z_g) \geq \delta$, but the sharp-threshold argument does not yet apply. This Erdős–Rényi comparison provides a density-based connectivity criterion and does not constitute a deterministic connectivity guarantee for arbitrary inhomogeneous graphon samples.*

Proof. Since $\eta(x) = \max(0, x) \geq 0$, the sum

$$\mathcal{L}_{\text{conn}} = \eta\left(\frac{1}{\bar{W}_n(z_g)n + \xi} - \rho(z_g)\right) + \eta\big(\delta - \bar{W}_n(z_g)\big)$$

vanishes if and only if both summands vanish individually. Hence $\mathcal{L}_{\text{conn}} = 0$ implies

$$\rho(z_g) \geq \frac{1}{\bar{W}_n(z_g)\,n + \xi}, \tag{66}$$

$$\bar{W}_n(z_g) \geq \delta. \tag{67}$$

Eq. 67 is precisely the bound in Eq. 64; unlike the case where the density term is evaluated on the unsparsified surface $\bar{W}_\psi(z_g)$, no further manipulation of Eq. 66 is needed or possible here, since $\bar{W}_\psi(z_g)$ does not appear in either inequality.

For the corollary on $\rho(z_g)$, each unsparsified graphon evaluation satisfies $W_\psi(s_i, s_j \mid z_g) \in (0,1)$, since it is obtained through a sigmoid output. Therefore, its mean satisfies

$$\bar{W}_\psi(z_g) = \frac{1}{|\mathcal{S}|} \sum_{(i,j)\in\mathcal{S}} W_\psi(s_i, s_j \mid z_g) \in (0,1). \tag{68}$$

Using the identity $\bar{W}_n(z_g) = \bar{W}_\psi(z_g)\rho(z_g)$, we then obtain

$$\rho(z_g) = \frac{\bar{W}_n(z_g)}{\bar{W}_\psi(z_g)} \geq \bar{W}_n(z_g) \geq \delta, \tag{69}$$

which is Eq. 65.

We also note that Eq. 66 is non-binding whenever Eq. 67 holds with $\bar{W}_n(z_g)\,n \gg 1$: since $\bar{W}_n(z_g) \geq \delta$ is bounded away from zero independently of n, the right-hand side $1/(\bar{W}_n(z_g)n + \xi)$ decays as

$\Theta(1/n)$ and is $\leq \delta \leq \rho(z_g)$ for all n satisfying $\delta n \gtrsim 1/\delta$, i.e. for essentially every n of practical interest; the density-floor term in Eq. 67 is therefore the binding constraint in the regime relevant to our datasets.

For the connectivity claim: let $p := \bar{W}_n(z_g) \geq \delta$. Since δ is a fixed constant independent of n, we have $pn \geq \delta n \to \infty$ as $n \to \infty$, which exceeds any fixed constant threshold c/n ($c > 1$) required for a giant component of size $(1-x)n + o(n)$ under the classical Erdős–Rényi double-jump result (Newman, 2002), where $x \in (0,1)$ is the unique root of $xe^{-c(1-x)} = x$. The guarantee is in fact stronger than a giant component alone: the sharp connectivity threshold for $\mathbb{G}(n,p)$ is $p^\star(n) = \ln n/n \to 0$, so once n is large enough that $\ln n/n < \delta$–numerically, $n \geq 20$ already gives $\ln n/n \approx 0.150 < \delta = 0.15$, and the margin widens rapidly for larger n–the classical sharp-threshold theorem gives

$$\Pr\left[\mathbb{G}(n, \bar{W}_n(z_g)) \text{ is connected}\right] \longrightarrow 1 \qquad \text{as } n \to \infty. \tag{70}$$

This subsumes the giant-component conclusion for all n in this regime. For smaller n where $\ln n/n \geq \delta$, only the density-floor bound $\bar{W}_n(z_g) \geq \delta$ of Eq. 64 is guaranteed for the learned graphon; empirically this still substantially reduces the probability of disconnected source graphs relative to an unregularised graphon, though it does not by itself force connectivity in that regime. □

Implication for the Stage-1 connectivity objective. Theorem 5 shows that $\mathcal{L}_{\text{conn}}(z_g, n) = 0$ implies the density floor $\bar{W}_n(z_g) \geq \delta$. Once n is sufficiently large that $\log n/n < \delta$, this lower bound lies above the classical Erdős–Rényi connectivity threshold, yielding connectivity probability approaching one under the corresponding homogeneous Erdős–Rényi comparison.

Lemma 6 (Laplacian perturbation bound)**.** *For symmetric $A, B \in \mathbb{R}^{n\times n}$ with zero diagonal,*

$$\|L(A) - L(B)\|_2 \;\leq\; (\sqrt{n}+1)\,\|A-B\|_2 \;\leq\; (\sqrt{n}+1)\,\|A-B\|_F.$$

Proof. Let $C = A - B$, which is symmetric with zero diagonal since A and B are. Write $L(A) - L(B) = \big(\operatorname{diag}(A\mathbf{1}) - A\big) - \big(\operatorname{diag}(B\mathbf{1}) - B\big) = \operatorname{diag}(C\mathbf{1}) - C$, where $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector. Denote $D_C := \operatorname{diag}(C\mathbf{1})$.

For any unit vector e_i of the standard basis, the i-th diagonal entry of D_C satisfies

$$|(D_C)_{ii}| = |e_i^\top C\mathbf{1}| \;\leq\; \|C\|_2\,\|e_i\|_2\,\|\mathbf{1}\|_2 \;=\; \sqrt{n}\,\|C\|_2,$$

by Cauchy–Schwarz applied to $e_i^\top C\mathbf{1} = (Ce_i)^\top \mathbf{1}$ combined with $\|Ce_i\|_2 \leq \|C\|_2$. Since D_C is diagonal, its operator norm equals the largest absolute diagonal entry, so

$$\|D_C\|_2 = \max_i |(D_C)_{ii}| \;\leq\; \sqrt{n}\,\|C\|_2.$$

By the triangle inequality for the operator norm,

$$\|L(A) - L(B)\|_2 = \|D_C - C\|_2 \;\leq\; \|D_C\|_2 + \|C\|_2 \;\leq\; \sqrt{n}\,\|C\|_2 + \|C\|_2 = (\sqrt{n}+1)\|C\|_2.$$

The second inequality in the statement follows from the standard fact that the operator (spectral) norm is dominated by the Frobenius norm, $\|C\|_2 \leq \|C\|_F$, since the Frobenius norm equals the ℓ_2-norm of the singular value vector while the operator norm is its maximum entry. □

Theorem 7 (Algebraic connectivity stability along the flow trajectory)**.** *Let A_t be any symmetric soft-adjacency path (e.g. the training-time rectified interpolation $A_t = (1-t)A_0 + tA_1$, or the generation-time Euler iterate) and suppose the terminal (or endpoint-predicted) adjacency satisfies $\lambda_2(L(A_1)) \geq \gamma > 0$ — the condition targeted by $\mathcal{L}_{\text{spec}}$ at γ margin. Then for every t,*

$$\lambda_2(L(A_t)) \;\geq\; \gamma - (\sqrt{n}+1)\,\|A_t - A_1\|_2.$$

In particular $L(A_t)$ has strictly positive algebraic connectivity whenever $\|A_t - A_1\|_2 < \gamma/(\sqrt{n}+1)$; for the rectified path $A_t = (1-t)A_0 + tA_1$ this becomes the sufficient condition $(1-t)\,\|A_0 - A_1\|_2 < \gamma/(\sqrt{n}+1)$.

Proof. Both $L(A_t)$ and $L(A_1)$ are real symmetric matrices, so Weyl's inequality applies: for any two symmetric matrices $M, N \in \mathbb{R}^{n\times n}$ and any eigenvalue index k,

$$|\lambda_k(M) - \lambda_k(N)| \;\leq\; \|M - N\|_2,$$

where eigenvalues are indexed in nondecreasing order. Applying this with $k = 2$, $M = L(A_t)$, and $N = L(A_1)$ gives

$$\lambda_2(L(A_t)) \;\geq\; \lambda_2(L(A_1)) - \|L(A_t) - L(A_1)\|_2.$$

By Lemma 6 (noting A_t and A_1 are symmetric with zero diagonal),

$$\|L(A_t) - L(A_1)\|_2 \;\leq\; (\sqrt{n}+1)\,\|A_t - A_1\|_2.$$

Combining the two displays and using the hypothesis $\lambda_2(L(A_1)) \geq \gamma$,

$$\lambda_2(L(A_t)) \;\geq\; \gamma - (\sqrt{n}+1)\,\|A_t - A_1\|_2,$$

which is the claimed bound. This right-hand side is strictly positive precisely when $\|A_t - A_1\|_2 < \gamma/(\sqrt{n}+1)$, giving the first corollary.

For the rectified path $A_t = (1-t)A_0 + tA_1$, we have $A_t - A_1 = (1-t)A_0 + tA_1 - A_1 = (1-t)(A_0 - A_1)$. Since $1 - t \geq 0$ is a nonnegative scalar, it factors out of the operator norm exactly:

$$\|A_t - A_1\|_2 = (1-t)\,\|A_0 - A_1\|_2.$$

Substituting into the positivity condition above gives the stated sufficient condition $(1-t)\|A_0 - A_1\|_2 < \gamma/(\sqrt{n}+1)$. □

Implication for connectivity along the transport trajectory. Lemma 6 and Theorem 7 establish a sufficient condition under which endpoint connectivity is retained along the interpolation path. In particular, if $\lambda_2(L(A_1)) \geq \gamma > 0$, then an intermediate adjacency A_t remains connected whenever $(1-t)\|A_0 - A_1\|_2$ lies below the threshold specified in Theorem 7, which depends on γ and the graph size n.

Assumption 1 (ℓ_1-Lipschitz target graphon)**.** *The (idealized, infinite-capacity) target graphon $W : [0,1]^2 \to [0,1]$ satisfies $|W(x,y) - W(x',y')| \leq L\,(|x-x'| + |y-y'|)$ for all $(x,y), (x',y') \in [0,1]^2$.*

Proposition 8 (Non-asymptotic Bernstein approximation rate)**.** *Under Assumption 1, let $\widehat{W}_K$ be the degree-K symmetrized Bernstein approximation of W (Eq. 21–24). Then*

$$\sup_{(x,y)\in[0,1]^2} \left|W(x,y) - \widehat{W}_K(x,y)\right| \;\leq\; \frac{L}{\sqrt{K}}.$$

Proof. Recall the (unsymmetrized) degree-K Bernstein approximation

$$\widetilde{W}_K(x,y) = \sum_{r=0}^{K}\sum_{s=0}^{K} W\left(\frac{r}{K}, \frac{s}{K}\right)\phi_{rs}^{(K)}(x,y), \qquad \phi_{rs}^{(K)}(x,y) = \binom{K}{r}x^r(1-x)^{K-r}\binom{K}{s}y^s(1-y)^{K-s},$$

and the symmetrization $\widehat{W}_K(x,y) = \frac{1}{2}\big(\widetilde{W}_K(x,y) + \widetilde{W}_K(y,x)\big)$.

Step 1: bound the unsymmetrized approximation. Fix $(x,y) \in [0,1]^2$ and let $X \sim \mathrm{Binomial}(K,x)$, $Y \sim \mathrm{Binomial}(K,y)$ be independent random variables. Since $\phi_{rs}^{(K)}(x,y) = \Pr(X=r)\Pr(Y=s)$, we may write

$$\widetilde{W}_K(x,y) = \mathbb{E}\left[W\left(\frac{X}{K}, \frac{Y}{K}\right)\right].$$

By Assumption 1 applied pointwise and Jensen's inequality (moving the absolute value inside the expectation),

$$\begin{aligned}\left|W(x,y) - \widetilde{W}_K(x,y)\right| &= \left|\mathbb{E}\left[W(x,y) - W\left(\frac{X}{K}, \frac{Y}{K}\right)\right]\right| \leq \mathbb{E}\left[L\left(\left|\frac{X}{K} - x\right| + \left|\frac{Y}{K} - y\right|\right)\right] \\ &= L\left(\mathbb{E}\left|\frac{X}{K} - x\right| + \mathbb{E}\left|\frac{Y}{K} - y\right|\right).\end{aligned} \tag{71}$$

By Cauchy–Schwarz (equivalently, Jensen applied to $|\cdot|$ under the square root),

$$\mathbb{E}\left|\frac{X}{K} - x\right| \;\leq\; \sqrt{\mathrm{Var}\left(\frac{X}{K}\right)} = \sqrt{\frac{x(1-x)}{K}} \;\leq\; \sqrt{\frac{1}{4K}} = \frac{1}{2\sqrt{K}},$$

using $x(1-x) \le \frac{1}{4}$ for $x \in [0,1]$, and identically $\mathbb{E}|Y/K - y| \le \frac{1}{2\sqrt{K}}$. Therefore

$$\left|W(x,y) - \widetilde{W}_K(x,y)\right| \;\le\; L\left(\frac{1}{2\sqrt{K}} + \frac{1}{2\sqrt{K}}\right) = \frac{L}{\sqrt{K}}.$$

Since (x,y) was arbitrary, $\sup_{(x,y)} |W(x,y) - \widetilde{W}_K(x,y)| \le L/\sqrt{K}$.

Step 2: symmetrization does not worsen the bound. Using $W(x,y) = W(y,x)$ (graphon symmetry) and the triangle inequality,

$$\begin{aligned}\left|W(x,y) - \widehat{W}_K(x,y)\right| &= \tfrac{1}{2}\left|\left(W(x,y) - \widetilde{W}_K(x,y)\right) + \left(W(y,x) - \widetilde{W}_K(y,x)\right)\right| \\ &\le \tfrac{1}{2}\Big(\left|W(x,y) - \widetilde{W}_K(x,y)\right| + \left|W(y,x) - \widetilde{W}_K(y,x)\right|\Big).\end{aligned} \tag{72}$$

Both terms on the right are bounded by $L/\sqrt{K}$ from Step 1 (the bound holds uniformly over all points, including (y,x)), so

$$\left|W(x,y) - \widehat{W}_K(x,y)\right| \;\le\; \tfrac{1}{2}\left(\frac{L}{\sqrt{K}} + \frac{L}{\sqrt{K}}\right) = \frac{L}{\sqrt{K}}.$$

Taking the supremum over $(x,y) \in [0,1]^2$ completes the proof. □

Theorem 9 (ELBO decomposition)**.** *For every graph G with canonical form $\widehat{G} = C(G)$, canonical permutation $\pi^\star$, canonically reordered node latents $\hat{z}_i = z_{\pi^\star(i)}$, and associated deterministic canonical coordinates $s_{1:n}$,*

$$\begin{aligned}\log p_\Theta(\widehat{G}) \ge\; &\mathbb{E}_{q_\phi(z_g, \hat{z}_{1:n}|G)}\left[\log p_\Theta(\widehat{G} \mid z_g, \hat{z}_{1:n}, s_{1:n})\right] \\ &- \mathrm{KL}(q_\phi(z_g \mid G) \| p(z_g)) \\ &- \mathbb{E}_{q_\phi(z_g|G)}\left[\mathrm{KL}(q_\phi(\hat{z}_{1:n} \mid G) \| p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n}))\right].\end{aligned} \tag{73}$$

Here $q_\phi(\hat{z}_{1:n} \mid G)$ denotes the deterministic permutation of the node-level posterior induced by $\pi^\star$.

Proof. The encoder first processes G in its observed node ordering and defines

$$q_\phi(z_g, z_{1:n} \mid G) = q_\phi(z_g \mid G) \prod_{i=1}^{n} q_\phi(z_i \mid G).$$

The canonical permutation $\pi^\star$ is then applied to the node-level posterior variables, with

$$\hat{z}_i = z_{\pi^\star(i)}, \qquad q_\phi(\hat{z}_i \mid G) := q_\phi(z_{\pi^\star(i)} \mid G).$$

Since this is a deterministic permutation, the resulting posterior in canonical coordinates factorises as

$$q_\phi(z_g, \hat{z}_{1:n} \mid G) = q_\phi(z_g \mid G) q_\phi(\hat{z}_{1:n} \mid G).$$

The joint generative model over the canonical graph and its latent variables, conditioned on the deterministic canonical coordinates $s_{1:n}$, is

$$p_\Theta(\widehat{G}, z_g, \hat{z}_{1:n} \mid s_{1:n}) = p_\Theta(\widehat{G} \mid z_g, \hat{z}_{1:n}, s_{1:n})\, p(z_g)\, p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n}),$$

which is exactly the generative factorisation in Eq. 5. Throughout the proof, $s_{1:n}$ is a fixed deterministic quantity associated with the canonical representation and is therefore not integrated or marginalised over.

Step 1: Marginalise and reweight by the variational posterior. By definition,

$$p_\Theta(\widehat{G} \mid s_{1:n}) = \iint p_\Theta(\widehat{G}, z_g, \hat{z}_{1:n} \mid s_{1:n})\, dz_g\, d\hat{z}_{1:n}.$$

Multiplying and dividing by $q_\phi(z_g, \hat{z}_{1:n} \mid G)$ gives

$$p_\Theta(\widehat{G} \mid s_{1:n}) = \iint q_\phi(z_g, \hat{z}_{1:n} \mid G) \frac{p_\Theta(\widehat{G}, z_g, \hat{z}_{1:n} \mid s_{1:n})}{q_\phi(z_g, \hat{z}_{1:n} \mid G)} \, dz_g \, d\hat{z}_{1:n} \tag{74}$$

$$= \mathbb{E}_{q_\phi(z_g, \hat{z}_{1:n} \mid G)} \left[\frac{p_\Theta(\widehat{G}, z_g, \hat{z}_{1:n} \mid s_{1:n})}{q_\phi(z_g, \hat{z}_{1:n} \mid G)} \right]. \tag{75}$$

Since $s_{1:n}$ is deterministically associated with $\widehat{G}$, we suppress this fixed conditioning in the notation and write $p_\Theta(\widehat{G}) \equiv p_\Theta(\widehat{G} \mid s_{1:n})$ below.

Step 2: Apply Jensen's inequality. Since $\log(\cdot)$ is concave,

$$\log p_\Theta(\widehat{G}) \geq \mathbb{E}_{q_\phi(z_g, \hat{z}_{1:n} \mid G)} \left[\log \frac{p_\Theta(\widehat{G}, z_g, \hat{z}_{1:n} \mid s_{1:n})}{q_\phi(z_g, \hat{z}_{1:n} \mid G)} \right].$$

Step 3: Expand the integrand. Using the generative and variational factorisations,

$$\log \frac{p_\Theta(\widehat{G}, z_g, \hat{z}_{1:n} \mid s_{1:n})}{q_\phi(z_g, \hat{z}_{1:n} \mid G)} = \log p_\Theta(\widehat{G} \mid z_g, \hat{z}_{1:n}, s_{1:n}) + \log \frac{p(z_g)}{q_\phi(z_g \mid G)} + \log \frac{p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n})}{q_\phi(\hat{z}_{1:n} \mid G)}. \tag{76}$$

Taking expectations term-wise gives

$$\log p_\Theta(\widehat{G}) \geq \underbrace{\mathbb{E}_{q_\phi(z_g, \hat{z}_{1:n} \mid G)} \left[\log p_\Theta(\widehat{G} \mid z_g, \hat{z}_{1:n}, s_{1:n}) \right]}_{\text{(I)}} + \underbrace{\mathbb{E}_{q_\phi(z_g \mid G)} \left[\log \frac{p(z_g)}{q_\phi(z_g \mid G)} \right]}_{\text{(II)}} + \underbrace{\mathbb{E}_{q_\phi(z_g, \hat{z}_{1:n} \mid G)} \left[\log \frac{p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n})}{q_\phi(\hat{z}_{1:n} \mid G)} \right]}_{\text{(III)}}. \tag{77}$$

Step 4: Simplify term (II). By the definition of the KL divergence,

$$\text{(II)} = -\mathrm{KL}(q_\phi(z_g \mid G) \| p(z_g)).$$

Step 5: Simplify term (III). Using the factorisation of the variational posterior and the tower property of expectation,

$$\text{(III)} = \mathbb{E}_{q_\phi(z_g \mid G)} \left[\mathbb{E}_{q_\phi(\hat{z}_{1:n} \mid G)} \left[\log \frac{p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n})}{q_\phi(\hat{z}_{1:n} \mid G)} \right] \right]. \tag{78}$$

For each fixed z_g, the inner expectation is the negative KL divergence,

$$\mathbb{E}_{q_\phi(\hat{z}_{1:n} \mid G)} \left[\log \frac{p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n})}{q_\phi(\hat{z}_{1:n} \mid G)} \right] = -\mathrm{KL}(q_\phi(\hat{z}_{1:n} \mid G) \| p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n})).$$

Hence

$$\text{(III)} = -\mathbb{E}_{q_\phi(z_g \mid G)} \left[\mathrm{KL}(q_\phi(\hat{z}_{1:n} \mid G) \| p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n})) \right].$$

Step 6: Combine. Substituting the expressions for (I), (II), and (III) gives

$$\log p_\Theta(\widehat{G}) \geq \mathbb{E}_{q_\phi(z_g, \hat{z}_{1:n} \mid G)} \left[\log p_\Theta(\widehat{G} \mid z_g, \hat{z}_{1:n}, s_{1:n}) \right] - \mathrm{KL}(q_\phi(z_g \mid G) \| p(z_g)) - \mathbb{E}_{q_\phi(z_g \mid G)} \left[\mathrm{KL}(q_\phi(\hat{z}_{1:n} \mid G) \| p_\omega(\hat{z}_{1:n} \mid z_g, s_{1:n})) \right], \tag{79}$$

which is the claimed bound. □

Assumption 2 (Lipschitz velocity field)**.** *$v_\theta(\cdot, t, z_g, z_{1:n})$ is L_v-Lipschitz in its graph argument, uniformly in $t \in [0, 1]$.*

Theorem 10 (Error propagation along Euler flow trajectories)**.** *Under Assumption 2, let $G(t)$ solve $\dot{G} = v_\theta(G, t)$ exactly and let $G_0, G_1, \ldots, G_K$ be the explicit Euler iterates of Algorithm 2 ($t_k = k/K$, $G_{k+1} = G_k + \frac{1}{K} v_\theta(G_k, t_k)$), with G_0 shared as the common initial condition for both the exact trajectory and the Euler iterates. If the local truncation error obeys*

$$\left\| G(t_{k+1}) - \left(G(t_k) + \tfrac{1}{K} v_\theta(G(t_k), t_k) \right) \right\| \;\le\; \tau_k / K^2$$

for constants τ_k (finite whenever $t \mapsto v_\theta(G(t), t)$ is continuously differentiable along the true trajectory), then the error $e_k := \|G_k - G(t_k)\|$ satisfies the recursion

$$e_{k+1} \le \left(1 + \frac{L_v}{K}\right) e_k + \frac{\tau_k}{K^2},$$

and hence

$$e_K \;\le\; \frac{1}{K^2} \sum_{k=0}^{K-1} \left(1 + \frac{L_v}{K}\right)^{K-1-k} \tau_k.$$

In particular, an error τ_k incurred at an early *step k (small t_k, close to the source G_0) is amplified by a factor $(1 + L_v/K)^{K-1-k}$ that can be as large as $\approx e^{L_v}$, whereas an error incurred at the final step $k = K - 1$ (close to the target G_1) is amplified by a factor of exactly* 1.

Proof. **Step 1: Initial error is zero.** Both the exact trajectory $G(t)$ and the Euler iterates $\{G_k\}$ start from the same graph G_0 (Algorithm 3, line 8, and the corresponding initialisation in Algorithm 2), so

$$e_0 := \|G_0 - G(t_0)\| = \|G_0 - G_0\| = 0.$$

Step 2: One-step error recursion. Fix $k \in \{0, \ldots, K-1\}$. By definition of the Euler update,

$$G_{k+1} = G_k + \tfrac{1}{K} v_\theta(G_k, t_k).$$

We compare this to the exact trajectory evaluated at t_{k+1} by inserting and subtracting the quantity $G(t_k) + \frac{1}{K} v_\theta(G(t_k), t_k)$ — the one-step Euler update applied to the exact solution at time t_k:

$$\begin{aligned} G_{k+1} - G(t_{k+1}) &= \left(G_k + \tfrac{1}{K} v_\theta(G_k, t_k) \right) - G(t_{k+1}) \\ &= \underbrace{\left(G_k - G(t_k) \right)}_{\text{(a) carried-over error}} + \underbrace{\tfrac{1}{K} \left(v_\theta(G_k, t_k) - v_\theta(G(t_k), t_k) \right)}_{\text{(b) velocity mismatch}} \\ &\quad - \underbrace{\left(G(t_{k+1}) - G(t_k) - \tfrac{1}{K} v_\theta(G(t_k), t_k) \right)}_{\text{(c) local truncation error}}. \end{aligned} \tag{80}$$

This is an algebraic identity: adding and subtracting the same quantity does not change the left-hand side, and regrouping the six resulting terms into (a), (b), −(c) reproduces exactly $G_{k+1} - G(t_{k+1})$, as can be verified by expanding all three brackets and cancelling $G(t_k)$ and $\frac{1}{K} v_\theta(G(t_k), t_k)$.

Step 3: Bound each term. Taking norms of Eq. 80 and applying the triangle inequality,

$$\|G_{k+1} - G(t_{k+1})\| \;\le\; \underbrace{\|G_k - G(t_k)\|}_{\text{(a)}} + \underbrace{\tfrac{1}{K} \left\| v_\theta(G_k, t_k) - v_\theta(G(t_k), t_k) \right\|}_{\text{(b)}} \tag{81}$$

$$+ \underbrace{\left\| G(t_{k+1}) - G(t_k) - \tfrac{1}{K} v_\theta(G(t_k), t_k) \right\|}_{\text{(c)}}. \tag{82}$$

Term (a) is, by definition, e_k.

Term (b): by Assumption 2, $v_\theta(\cdot, t_k)$ is L_v-Lipschitz in its graph argument (uniformly in t, hence in particular at $t = t_k$), so

$$\left\| v_\theta(G_k, t_k) - v_\theta(G(t_k), t_k) \right\| \;\le\; L_v \, \|G_k - G(t_k)\| = L_v \, e_k,$$

giving term (b) $\le \frac{L_v}{K} e_k$.

Term (c) is exactly the local truncation error, bounded by hypothesis:

$$\left\| G(t_{k+1}) - G(t_k) - \tfrac{1}{K} v_\theta(G(t_k), t_k) \right\| \;\le\; \frac{\tau_k}{K^2}.$$

Combining the three bounds,

$$e_{k+1} \;\le\; e_k + \frac{L_v}{K} e_k + \frac{\tau_k}{K^2} = \Big(1 + \frac{L_v}{K}\Big) e_k + \frac{\tau_k}{K^2},$$

which is the claimed one-step recursion.

Step 4: Unroll the recursion by induction. We claim that for every $m = 0, 1, \ldots, K$,

$$e_m \;\le\; \frac{1}{K^2} \sum_{k=0}^{m-1} \Big(1 + \frac{L_v}{K}\Big)^{m-1-k} \tau_k \qquad \text{(with the empty sum for } m = 0 \text{ interpreted as } 0).$$

Base case ($m = 0$). The right-hand side is the empty sum, equal to 0, matching $e_0 = 0$ from Step 1.

Inductive step. Suppose the claim holds for some $m < K$, i.e. $e_m \le \frac{1}{K^2} \sum_{k=0}^{m-1} (1+L_v/K)^{m-1-k} \tau_k$. Applying the one-step recursion from Step 3 with $k = m$,

$$e_{m+1} \;\le\; \Big(1 + \frac{L_v}{K}\Big) e_m + \frac{\tau_m}{K^2}.$$

Substituting the inductive hypothesis for e_m,

$$\begin{aligned} e_{m+1} &\le \Big(1 + \frac{L_v}{K}\Big) \left[\frac{1}{K^2} \sum_{k=0}^{m-1} \Big(1 + \frac{L_v}{K}\Big)^{m-1-k} \tau_k \right] + \frac{\tau_m}{K^2} \\ &= \frac{1}{K^2} \sum_{k=0}^{m-1} \Big(1 + \frac{L_v}{K}\Big)^{m-k} \tau_k \;+\; \frac{1}{K^2} \Big(1 + \frac{L_v}{K}\Big)^{0} \tau_m \\ &= \frac{1}{K^2} \sum_{k=0}^{m} \Big(1 + \frac{L_v}{K}\Big)^{m-k} \tau_k, \end{aligned}$$

where the middle line multiplies $(1 + L_v/K)$ into the sum (raising each exponent $m - 1 - k$ to $m - k$) and rewrites the added term τ_m/K^2 as the $k = m$ term of the sum with exponent $(1 + L_v/K)^{m-m} = 1$. This is exactly the claimed bound for $m + 1$ (with $m - k$ playing the role of $(m + 1) - 1 - k$), completing the induction.

Step 5: Evaluate at $m = K$. Setting $m = K$ in the Step 4 claim gives

$$e_K \;\le\; \frac{1}{K^2} \sum_{k=0}^{K-1} \Big(1 + \frac{L_v}{K}\Big)^{K-1-k} \tau_k,$$

which is the claimed global error bound.

Step 6: Amplification-factor interpretation. Consider the weight $w_k := (1 + L_v/K)^{K-1-k}$ multiplying τ_k/K^2 in the sum above, as a function of the step index k (equivalently, of $t_k = k/K$). As $K \to \infty$ with $t_k = k/K$ held fixed,

$$\Big(1 + \frac{L_v}{K}\Big)^{K-1-k} = \Big(1 + \frac{L_v}{K}\Big)^{K(1-t_k)-1} \;\longrightarrow\; e^{L_v(1-t_k)},$$

using the standard limit $(1 + x/K)^K \to e^x$. At the earliest step, $k = 0$ (so $t_k = 0$), this weight tends to e^{L_v}: an error made at the very start of the trajectory is compounded through all $K - 1$ remaining Lipschitz-bounded Euler steps. At the final step, $k = K - 1$ (so $t_k \to 1$), the weight is $(1 + L_v/K)^0 = 1$ exactly, independent of K: an error made at the last step has no subsequent steps left to compound through, and enters the global error bound unamplified. This asymmetry is monotonic in k, since w_k is a decreasing function of k for fixed K (the exponent $K - 1 - k$ strictly decreases as k increases), confirming that early-trajectory truncation errors are weighted at least as heavily as any later error, with the extreme cases $\approx e^{L_v}$ and exactly 1 as stated. □

Algorithm 1 Lanczos estimation of algebraic connectivity $\lambda_2(\tilde{L}_1)$

Require: Soft adjacency $\tilde{A}_1 \in [0,1]^{n\times n}$; number of Lanczos steps k
Ensure: Estimate $\hat{\lambda}_2 \approx \lambda_2(\tilde{L}_1)$, differentiable w.r.t. $\tilde{A}_1$

if $n < 2$ **then**
  return 0 ▷ connectivity undefined/trivial for $n < 2$
end if
$k \leftarrow \min(k, n-1)$ ▷ cap steps to available non-trivial dimensions
$d \leftarrow \tilde{A}_1\mathbf{1}$; $\tilde{L}_1 \leftarrow \mathrm{diag}(d) - \tilde{A}_1$
$\mathbf{u} \leftarrow \mathbf{1}/\sqrt{n}$ ▷ trivial eigenvector of $\tilde{L}_1$
Sample $v_1 \sim \mathcal{N}(0, I_n)$; $v_1 \leftarrow v_1 - (v_1^\top \mathbf{u})\mathbf{u}$; $v_1 \leftarrow v_1/\|v_1\|$
$\beta_0 \leftarrow 0$; $v_0 \leftarrow \mathbf{0}$
for $j = 1, \ldots, k$ **do**
  $w_j \leftarrow \tilde{L}_1 v_j$ ▷ $O(n)$ for sparse $\tilde{L}_1$
  $w_j \leftarrow w_j - (w_j^\top \mathbf{u})\mathbf{u}$ ▷ re-deflate against trivial eigenvector
  $\alpha_j \leftarrow v_j^\top w_j$
  $w_j \leftarrow w_j - \alpha_j v_j$
  if $j > 1$ **then**
    $w_j \leftarrow w_j - \beta_{j-1} v_{j-1}$
  end if
  $\beta_j \leftarrow \|w_j\|$
  $v_{j+1} \leftarrow w_j/\beta_j$ ▷ only v_j, v_{j-1} retained; memory $O(n+k)$
end for
$T \leftarrow \mathrm{tridiag}(\beta_{1:k-1}, \alpha_{1:k}, \beta_{1:k-1}) \in \mathbb{R}^{k\times k}$
$\hat{\lambda}_2 \leftarrow \lambda_{\min}(T)$ ▷ dense eigendecomposition, $O(k^3)$
return $\hat{\lambda}_2$ ▷ overall: $O(kn)$ time, $O(n+k)$ memory

D Algorithms

E Extended Background

This appendix gives the full technical background underlying Section 2, including the canonical-labelling procedure (Section E.1, supporting Section 2.1), Flow Matching (Section E.2, supporting Section 2.2), graphons and their sparse extensions (Section E.3, supporting Section 2.3), and the algebraic-connectivity construction (Section E.4, supporting Section 2.4).

E.1 Canonical Labelling via NAUTY

A fundamental property of graph data is that a fixed matrix representation is *not permutation invariant*: i.e., the same underlying graph can be represented by different adjacency and feature matrices under different vertex orderings. In what follows, we require permutation-invariant graph representations so that isomorphic graphs receive identical encodings, predictions, and likelihoods regardless of the arbitrary ordering of their vertices. Formally, two graphs $G = (A, X, E)$ and $G' = (A', X', E')$ are *isomorphic*, denoted by $G \cong G'$, if there exists a permutation matrix $P \in \{0,1\}^{n\times n}$, containing exactly one non-zero entry in every row and every column, together with the corresponding permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, such that

$$A' = PAP^\top, \qquad X' = PX, \qquad E'_{ij} = E_{\pi(i)\pi(j)}, \quad \forall\, i, j \in \{1, \ldots, n\}. \tag{83}$$

Eq. 83 states that G' is obtained solely by relabelling the vertices of G; although the adjacency and feature matrices change, the underlying graph structure remains identical.

This permutation symmetry presents a fundamental challenge for graph learning (Section 2.1). Since many different matrix representations correspond to the same graph, learning objectives defined directly on adjacency matrices may assign different supervision signals to structurally identical graphs. Consequently, optimization becomes dependent on arbitrary node orderings rather than the intrinsic graph topology. This issue is particularly problematic for graphon learning, where each

Algorithm 2 GraphonFlow training

Require: Training graphs $\mathcal{D}$; VAE q_ϕ with node prior p_ω and attribute decoders f_X, f_E; graphon W_ψ; velocity field v_θ; hyperparameters

Stage 1: Probabilistic structural learning
for epoch $= 1, \dots, E_1$ **do**
  for each graph $G_1 = (A_1, X_1, E_1) \in \mathcal{D}$ **do**
    $(\mu_g, \log \sigma_g^2), (\mu_n, \log \sigma_n^2) \leftarrow q_\phi(G_1)$
    $z_g = \mu_g + \sigma_g \odot \epsilon_g, \quad z_i = \mu_{n,i} + \sigma_{n,i} \odot \epsilon_i$ ▷ $\epsilon_g, \epsilon_i \sim \mathcal{N}(0, I)$
    $\pi^* \leftarrow \textsc{CanonicalLabelling}(A_1, X_1)$
    Canonicalise $\hat{A}_1, \hat{X}_1, \hat{E}_1$ and reorder node posterior parameters/latents by π^*
    Set the fixed canonical lattice $s_{1:n} = \{s_i\}_{i=1}^n \subset [0, 1]$
    $(\mu_p, \log \sigma_p^2) \leftarrow p_\omega(\hat{z}_i \mid z_g, s_i)$
    $P_1 \leftarrow W_\psi(s_{1:n}, s_{1:n} \mid z_g)$
    Compute Stage-1 objective $\mathcal{L}_{S1}$ (Eq. 19)
    Update the Stage-1 parameters using $\nabla \mathcal{L}_{S1}$
    Gradient clipping: $\|\nabla\| \leq 1$
  end for
end for

Stage 2: Structure-preserving flow learning (freeze Stage-1 modules)
for epoch $= 1, \dots, E_2$ **do**
  for each graph $G_1 = (A_1, X_1, E_1) \in \mathcal{D}$ **do**
    $\pi^* \leftarrow \textsc{CanonicalLabelling}(A_1, X_1)$
    $(\hat{A}_1, \hat{X}_1, \hat{E}_1) \leftarrow (A_1, X_1, E_1)[\pi^*]$
    $(\mu_g, \log \sigma_g^2), (\mu_n, \log \sigma_n^2) \leftarrow q_\phi(G_1)$ ▷ frozen encoder
    $z_g = \mu_g + \sigma_g \odot \epsilon_g, \quad z_i = \mu_{n,i} + \sigma_{n,i} \odot \epsilon_i$
    $\hat{z}_i \leftarrow z_{\pi^*(i)}$
    Use the fixed canonical lattice $s_{1:n} = \{s_i\}_{i=1}^n \subset [0, 1]$
    $A_0 \sim \text{Bernoulli}(W_\psi(s_{1:n}, s_{1:n} \mid z_g))$
    $(X_0, E_0) \leftarrow (f_X, f_E)(\hat{z}_{1:n}, A_0)$
    $t \sim \text{Unif}[0, 1]$
    $(A_t, X_t, E_t) \leftarrow (1 - t)(A_0, X_0, E_0) + t(\hat{A}_1, \hat{X}_1, \hat{E}_1)$
    $(v_A, v_X, v_E) \leftarrow v_\theta(A_t, X_t, E_t, t, z_g, \hat{z}_{1:n})$
    Compute Stage-2 objective $\mathcal{L}_{S2}$ (Eq. 20)
    Update θ using $\nabla \mathcal{L}_{S2}$
    Gradient clipping: $\|\nabla\| \leq 1$
  end for
end for

matrix position (i, j) corresponds to a fixed location on a continuous graphon surface. If isomorphic graphs are presented with inconsistent node orderings, identical structural patterns are mapped to different graphon coordinates, resulting in contradictory supervision and degraded learning.

Graphons are not identifiable as pointwise functions: graphons related by measure-preserving transformations of $[0, 1]$ define the same exchangeable graph distribution (Diaconis & Janson, 2007; Lovász, 2012; Borgs et al., 2018). At the finite-graph level, this corresponds to the fact that different vertex orderings produce different adjacency matrices while representing the same underlying graph; without a consistent alignment mechanism, isomorphic training graphs induce different matrix- or function-valued supervision signals.

We resolve this using McKay's NAUTY algorithm (McKay, 1981; McKay & Piperno, 2014), one of the most widely used practical tools for graph isomorphism testing, automorphism-group computation, and canonical labelling. NAUTY follows the individualisation-and-refinement paradigm: it iteratively refines an ordered partition of the vertex set and, when refinement does not yield a discrete partition, individualises selected vertices and explores the resulting search tree to determine a canonical labelling. Although the refinement-and-search procedure can require an exponential, and in the worst case potentially factorial-sized, search tree, NAUTY is highly effective on many graph classes

Algorithm 3 GraphonFlow generation

Require: Trained graphon W_ψ; trained velocity field v_θ; trained decoders f_X, f_E; node count n; integration steps K

$z_g \sim \mathcal{N}(0, I)$ ▷ sample graph-level latent
$s_i \leftarrow \dfrac{i-1}{n-1}$ for $i = 1, \dots, n$ ▷ canonical lattice, Section 3.2
$(\mu_p, \log \sigma_p^2) \leftarrow p_\omega(z_i \mid z_g, s_i)$ ▷ graph- and position-conditioned node prior
$z_i \sim \mathcal{N}(\mu_p, \mathrm{diag}(\sigma_p^2))$ for $i = 1, \dots, n$
$A_0 \sim \mathrm{Bernoulli}(W_\psi(s_i, s_j \mid z_g))$
$X_{0,i} \leftarrow \psi\big(f_X(z_i)\big)$ for $i = 1, \dots, n$
$E_{0,ij} \leftarrow A_{0,ij} \cdot \psi\big(f_E([z_i \| z_j])\big)$
$G_0 \leftarrow (A_0, X_0, E_0)$
for $k = 0, \dots, K-1$ **do**
  $t_k \leftarrow k/K$
  $(v_A, v_X, v_E) \leftarrow v_\theta(G_{t_k}, t_k, z_g, \{z_i\}_{i=1}^n)$
  $G_{t_{k+1}} \leftarrow G_{t_k} + \frac{1}{K}(v_A, v_X, v_E)$
end for
$\tilde{A} \leftarrow \mathrm{softplus}\big(\frac{1}{2}(A_{t_K} + A_{t_K}^\top)\big)$
$\tilde{E} \leftarrow \frac{1}{2}(E_{t_K} + E_{t_K}^\top)$
$\hat{X} \leftarrow \mathrm{argmax}(X_{t_K})$; $\hat{E} \leftarrow \mathrm{argmax}(\tilde{E})$
return $\hat{G} = (\hat{A}, \hat{X}, \hat{E})$

encountered in practice (McKay, 1981; McKay & Piperno, 2014). This makes it computationally suitable for the relatively small molecular graphs considered in this work.

The canonicalisation procedure is complemented by vertex colouring, allowing domain-specific node semantics to be incorporated whenever informative node attributes are available. For molecular graphs, we assign each atom a colour according to its chemical identity, $\chi_i = (\mathrm{AtomType}_i, \mathrm{FormalCharge}_i)$, so that NAUTY considers only colour-preserving automorphisms. Consequently, atoms with different element types or formal charges cannot be permuted into one another, preserving chemically meaningful symmetries while reducing the set of admissible vertex relabellings. Specifically, if a molecule has n atoms partitioned into C colour classes of sizes $n_1, \dots, n_C$, with $\sum_{c=1}^{C} n_c = n$, then the unconstrained permutation space contains $n!$ possible vertex orderings, whereas colour-preserving permutations are restricted to at most $\prod_{c=1}^{C} n_c!$ orderings. Thus, colouring reduces the combinatorial search space by a factor of $n!/\prod_{c=1}^{C} n_c!$, which is the multinomial number of ways in which atoms from different colour classes could otherwise be interleaved. This reduction can be substantial for chemically diverse molecules and also prevents the learning procedure from treating chemically invalid relabellings as equivalent examples.

For synthetic graphs, the same framework uses available node attributes as vertex colours whenever they are present. For homogeneous or featureless graphs, all vertices are assigned a common colour, in which case canonicalisation is determined entirely by topology and the admissible permutation space remains $n!$ before structural refinement. Thus, the same canonical-labelling procedure applies uniformly to attributed molecular graphs and featureless synthetic graphs, with the colouring strategy adapting naturally to the available information. For the relatively small graphs considered in this work, canonicalisation remains efficient in practice.

In *GraphonFlow*, this property plays a central role. By performing graphon supervision on canonical graph representations rather than arbitrary adjacency matrices, every isomorphic graph produces an identical supervision target. Consequently, the graphon learns structural relationships between canonical graph positions instead of dataset-dependent node indices, completely removing permutation ambiguity from the graphon learning objective while preserving the expressive power of the latent graph representation.

E.2 Flow Matching

Flow Matching (FM) (Lipman et al., 2023; Liu et al., 2023) is a continuous-time generative modelling framework that learns a deterministic transport map between a simple source distribution and a target data distribution. Unlike diffusion models, which progressively remove stochastic noise through a sequence of denoising steps, FM directly learns the velocity field that continuously transports samples from the source distribution to the data distribution. This formulation enables efficient sampling by solving a deterministic Ordinary Differential Equation (ODE), often requiring substantially fewer function evaluations than diffusion-based methods while achieving comparable generation quality.

Unlike score-based diffusion, which estimates the gradient of a stochastic reverse-time process, FM directly regresses the deterministic transport velocity along the probability path defined by Eq. 1 (Section 2.2). Consequently, the learned vector field defines an ordinary differential equation

$$\frac{dG(t)}{dt} = v_\theta(G(t), t), \qquad G(0) \sim p_0, \tag{84}$$

whose solution transports samples from the source distribution towards the target distribution.

At inference time, generation proceeds by first sampling an initial graph $G(0) \sim p_0$, and then numerically integrating Eq. 84 from $t = 0$ to $t = 1$ using a standard ODE solver (e.g., explicit Euler or Runge–Kutta). The final state $G(1)$ constitutes the generated graph.

The effectiveness of FM critically depends on the choice of the source distribution p_0. Existing graph flow models typically employ independent Gaussian noise or heuristic graph initializations, requiring the learned velocity field to simultaneously recover global graph topology and refine local graph attributes throughout transport. In contrast, our method learns a latent-conditioned graphon prior that provides a structurally meaningful source distribution, allowing the velocity field to focus primarily on refining graph structures rather than reconstructing them from unstructured noise (Section 3.2).

E.3 Graphons as Structural Graph Priors

Since classical graphons are non-parametric, we adopt a differentiable basis-expansion parameterization,

$$W_\psi(x, y) = \sigma\left(\sum_{b=1}^{B} \theta_b f_b(x, y)\right), \tag{85}$$

detailed further in Section 3.1. Unlike an adjacency matrix, which represents a single graph realization, a graphon (Eq. 2, Section 2.3) represents an entire distribution over graphs and therefore captures global structural characteristics of a graph family. The graphon therefore defines a continuous probability surface over latent node positions, allowing structurally similar nodes to exhibit similar connectivity patterns.

Different choices of basis functions in Eq. 85 provide different approximation properties. The Fourier basis efficiently captures smooth global connectivity patterns through spectral representations, while the Bernstein polynomial basis provides a flexible non-negative approximation of continuous graphon surfaces.

A standard graphon with a fixed, non-vanishing mean edge probability generates a dense graph sequence. In particular, if

$$\overline{W} = \int_{[0,1]^2} W(x, y)\, dx\, dy > 0,$$

then the expected number of edges in an n-node graph sampled from W satisfies $\mathbb{E}[m] = \binom{n}{2}\overline{W} \in \Theta(n^2)$. Since many real-world networks are sparse, we adopt the size-dependent sparsification

$$W_n(x, y) = \rho_n W(x, y), \qquad \rho_n \to 0, \tag{86}$$

following standard practice for sparse inhomogeneous random graphs. Under this scaling, $\mathbb{E}[m] \in \Theta(\rho_n n^2)$ and $\mathbb{E}[\overline{d}] \in \Theta(\rho_n n)$, where $\overline{d}$ denotes the average degree. Thus, choosing $\rho_n = \Theta(1/n)$ yields bounded expected average degree.

This scaled-graphon construction is a useful model for sparse inhomogeneous random graphs, but represents only one approach to sparsity. More general models, such as graphex processes, naturally

generate sparse exchangeable graphs and can accommodate heavy-tailed degree distributions and other sparse network structures that cannot be captured solely through a global sparsity factor (Caron & Fox, 2017; Borgs et al., 2018).

Recent work has incorporated graphons and related structured objects into graph representation learning and generation (Xu et al., 2021; Xia et al., 2023; Wijesinghe et al., 2026). We contrast these approaches with our latent-conditioned construction in Appendix F.

E.4 Algebraic Graph Connectivity

Connectivity is a fundamental structural property: disconnected components correspond to chemically invalid molecules and can disrupt functional interactions in biological and social networks. Consider a predicted soft adjacency matrix $\hat{A} \in [0,1]^{n\times n}$, where $\hat{A}_{ij}$ denotes the predicted probability (or weight) of an edge between vertices i and j. The corresponding degree of node i is

$$d_i = \sum_{j=1}^{n} \hat{A}_{ij}, \qquad i = 1, \dots, n. \tag{87}$$

Collecting these degrees gives $d = (d_1, \dots, d_n)^\top = \hat{A}\mathbf{1}$, and hence $D = \mathrm{diag}(d)$. The unnormalised graph Laplacian is

$$L = D - \hat{A}. \tag{88}$$

The spectrum of L encodes global structural properties of the graph. In particular, its second-smallest eigenvalue $\lambda_2(L)$, known as the algebraic connectivity or Fiedler value, characterises connectivity:

$$\lambda_2(L) > 0 \iff G \text{ is connected.} \tag{89}$$

Larger values of $\lambda_2(L)$ generally indicate stronger algebraic connectivity, whereas values approaching zero indicate weak or disconnected structure.

Since *GraphonFlow* predicts continuous edge probabilities rather than binary edge indicators, the Laplacian is a differentiable function of the predicted adjacency matrix. We therefore construct the soft Laplacian

$$\hat{L} = \mathrm{diag}(\hat{A}\mathbf{1}) - \hat{A},$$

whose eigenvalues can be incorporated into the training objective through automatic differentiation. This differentiable spectral characterisation forms the basis of the Stage 2 connectivity-aware objective in Section 3.3, allowing connectivity to be promoted without discrete projection during training.

F Related Work

Graph generative models. Deep graph generation has been studied through autoregressive, latent-variable, and normalizing-flow formulations. Autoregressive methods decompose graph generation into sequential node and edge decisions: GraphRNN generates nodes and edges recurrently (You et al., 2018), while GRAN improves efficiency and long-range dependency modelling by generating blocks of nodes and edges with graph recurrent attention (Liao et al., 2019). In molecular generation, structured latent-variable approaches such as Junction Tree VAE generate chemically meaningful substructures before assembling the molecular graph (Jin et al., 2018), whereas flow-based models such as GraphAF and MoFlow learn invertible transformations for molecular graphs (Shi et al., 2020; Zang & Wang, 2020). These approaches established effective mechanisms for graph generation, but sequential generation can depend on node ordering and require many dependent generation decisions, while molecule-specific constructions do not directly provide a general mechanism for modelling global topology across graph domains. Graphons provide an alternative probabilistic description of graph structure (Lovász, 2012). Graphon Autoencoders couple graph representations with graphon-factorisation decoders (Xu et al., 2021), and Implicit Graphon Neural Representation parameterises graphons using neural functions and supports generation from graphs of different sizes (Xia et al., 2023). These works motivate graphons as expressive structural models, but do not use a latent-conditioned graphon as the learned source distribution of a continuous-time graph transport process.

Diffusion-based graph generation. Diffusion models provide a non-autoregressive alternative by learning to reverse a progressive corruption process. Early score-based graph generators formulate generation over continuous relaxations of graph structure: EDP-GNN learns permutation-invariant score functions over adjacency matrices (Niu et al., 2020), while GDSS jointly models node and edge distributions through a system of stochastic differential equations (Jo et al., 2022). GraphGDP similarly formulates permutation-equivariant graph generation through continuous-time diffusion (Huang et al., 2022). In contrast, DIGRESS operates directly on categorical node and edge states using discrete denoising diffusion and a graph transformer (Vignac et al., 2023), while autoregressive diffusion combines discrete diffusion with sequential graph reconstruction (Kong et al., 2023). More recent approaches further improve the treatment of graph topology and discrete structure. GruM models generation as a mixture of endpoint-conditioned diffusion processes to explicitly incorporate the final graph structure (Jo et al., 2024), and discrete-state continuous-time diffusion uses continuous-time Markov chains to retain discrete graph states while allowing flexible sampling schedules (Xu et al., 2024). HOG-Diff incorporates higher-order topology through a coarse-to-fine diffusion bridge, progressively refining higher-order structural skeletons into pairwise graph connectivity (Huang & Birdal, 2026). Collectively, these developments substantially improve graph generation, but their generative processes remain primarily formulated around reversing corruption from a generic noise distribution or traversing prescribed intermediate processes, rather than learning a latent-conditioned structured source distribution from the data.

Flow matching for graph generation. Flow matching provides a simulation-free objective for learning continuous-time vector fields that transport a source distribution to a target distribution (Lipman et al., 2023), while discrete flow formulations extend this principle to categorical state spaces through continuous-time Markov chains (Campbell et al., 2024; Gat et al., 2024). These developments have motivated several graph-specific flow models. DEFOG formulates graph generation as discrete flow matching and separates training from sampling, enabling flexible sampling strategies with substantially fewer refinement steps than many diffusion approaches (Qin et al., 2025). GGFLOW combines discrete flow matching with optimal-transport coupling and an edge-augmented graph transformer (Hou et al., 2024), while SIMGFM simplifies graph discrete flow matching through a scheduler-based velocity formulation and endpoint-focused sampling (Luo et al., 2026). In the continuous setting, FLOWETTE evolves relaxed adjacency, node, and edge representations with a GNN-based velocity field and introduces FGW-based source–target coupling together with graphette structural priors (Wijesinghe et al., 2026). Flow matching has also been developed extensively for three-dimensional molecular generation, where continuous geometric variables and discrete molecular attributes can be transported jointly (Dunn & Koes, 2026). More recently, graphons have been incorporated into deep generative modelling as structural priors. Flowette (Wijesinghe et al., 2026) introduced *graphettes*, which augment graphons with graph-editing operators to provide richer structural source distributions for flow matching. This demonstrates that initializing transport from structured graph priors can substantially simplify the learning problem compared with transporting from independent noise. However, existing graphon-based approaches typically employ fixed graphon parameterizations or optimize graphons independently of latent graph representations, limiting their ability to adapt to individual graph instances. These works demonstrate the flexibility of flow-based graph generation, but the source distribution is generally specified independently of a learned latent representation, or structural compatibility is introduced through coupling, guidance, or manually specified priors after the source family has been chosen.

Positioning of GraphonFlow. *GraphonFlow* lies at the intersection of latent graph representation learning, graphon-based structural modelling, and continuous flow matching. Unlike graphon autoencoders and implicit graphon representations (Xu et al., 2021; Xia et al., 2023), our goal is not only to reconstruct or estimate a graphon: the learned graphon also defines the structured source distribution from which transport begins. Unlike discrete graph flow models (Qin et al., 2025; Hou et al., 2024; Luo et al., 2026), *GraphonFlow* learns a continuous velocity field over relaxed attributed graphs; and unlike Flowette (Wijesinghe et al., 2026), it does not require a pre-specified graphon/graphette source together with pairwise FGW-based source–target coupling. Instead, a first stage learns a hybrid graph- and node-level latent representation and a latent-conditioned graphon, with canonical graph labelling providing consistent graphon coordinates; the resulting structural model is then frozen and used to generate structured sources for conditional flow matching. Thus, *GraphonFlow* shifts structural modelling from an independently specified source or transport-time alignment mechanism to a learned, latent-conditioned structured source distribution, allowing the

subsequent flow to refine already structured graphs rather than recover topology entirely from unstructured noise.

G Additional Architectural and Derivation Details

This appendix expands on architectural and derivation details deferred from Section 3: the encoder (Appendix G.1), the design rationale for the latent-graphon decomposition (Appendix G.3), the conditional velocity field (Appendix G.4), and the connectivity regularisers of Section 3.3 (Appendix G.6).

G.1 Encoder Architecture

The encoder consists of L_{enc} stacked edge-aware GINE layers (Xu et al., 2019). Let $h_i^{(\ell)}$ denote the representation of node i at layer ℓ, and let e_{ji} denote the attributes of edge (j, i). Each GINE layer performs the update

$$h_i^{(\ell+1)} = \text{MLP}^{(\ell)}\left((1+\epsilon^{(\ell)})h_i^{(\ell)} + \sum_{j\in\mathcal{N}(i)} \text{ReLU}\left(h_j^{(\ell)} + W_e^{(\ell)} e_{ji}\right)\right), \tag{90}$$

where $\mathcal{N}(i)$ denotes the neighbourhood of node i, $\epsilon^{(\ell)}$ is the GINE self-feature weighting parameter, and $W_e^{(\ell)}$ projects edge attributes to the node hidden dimension when required. We denote the final node representation by $h_i = h_i^{(L_{\text{enc}})} \in \mathbb{R}^{d_h}$.

A graph-level representation is constructed by concatenating permutation-invariant mean and sum pooling,

$$h_G = [h_G^{\text{mean}} \| h_G^{\text{sum}}], \qquad h_G^{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} h_i, \qquad h_G^{\text{sum}} = \sum_{i=1}^{n} h_i. \tag{91}$$

Although $h_G^{\text{sum}} = n h_G^{\text{mean}}$, the two readouts provide complementary information when graph sizes vary: mean pooling captures size-normalised feature statistics, whereas sum pooling retains an explicit dependence on graph cardinality. Since GINE message passing is permutation equivariant and both pooling operators are permutation invariant, h_G is invariant to node permutations, whereas the collection of node representations $\{h_i\}_{i=1}^n$ is permutation equivariant. This is the readout used in the permutation-invariance argument of Theorem 3 (Appendix C), which cites Eq. 91 directly.

The graph-level posterior parameters are predicted from the pooled representation as

$$(\mu_g, \log \sigma_g^2) = f_g(h_G), \tag{92}$$

while the node-level posterior parameters are obtained independently from the corresponding node representations,

$$(\mu_{n,i}, \log \sigma_{n,i}^2) = f_n(h_i), \qquad i = 1, \dots, n, \tag{93}$$

where f_g and f_n are MLPs. Consequently, the graph-level posterior $q_\phi(z_g \mid G_1)$ is invariant to node permutations, while the collection of node-level posteriors $\{q_\phi(z_i \mid G_1)\}_{i=1}^n$ is permutation equivariant. This permits the encoder to operate on G_1 in its observed ordering, after which the sampled node-level latents can be placed into canonical order using the NAUTY permutation $\pi^\star$, as described in Section 3.1.

Predicting the log-variance provides an unconstrained parameterisation in $\mathbb{R}$, which is transformed to a positive variance when constructing the Gaussian posteriors in Eq. 4. Latent samples are obtained using the standard reparameterisation estimator.

Reparameterised sampling. For a Monte Carlo sample s, we draw $\epsilon_g^{(s)}, \epsilon_i^{(s)} \sim \mathcal{N}(0, I)$ and set

$$z_g^{(s)} = \mu_g + \sigma_g \odot \epsilon_g^{(s)}, \qquad z_i^{(s)} = \mu_{n,i} + \sigma_{n,i} \odot \epsilon_i^{(s)}. \tag{94}$$

These samples provide the graph- and node-level latent variables used by the probabilistic decoder in Section 3.1.

G.2 Attribute Decoder and Reconstruction Objective

For molecular graphs, node and bond attributes are categorical. Let $f_X : \mathbb{R}^{d_n} \to \mathbb{R}^{d_X}$ denote the node-attribute decoder and $f_E : \mathbb{R}^{2d_n} \to \mathbb{R}^{d_E}$ the edge-attribute decoder. The node decoder defines $\mathbf{p}_i^X(z_i) = \mathrm{softmax}(f_X(z_i))$. For a node pair (i, j), we enforce an undirected edge decoder by symmetrising the logits, $\boldsymbol{\ell}_{ij}^E(z_i, z_j) = \frac{1}{2}\left[f_E([z_i \| z_j]) + f_E([z_j \| z_i])\right]$, giving $\mathbf{p}_{ij}^E(z_i, z_j) = \mathrm{softmax}\big(\boldsymbol{\ell}_{ij}^E(z_i, z_j)\big)$, which satisfies $\mathbf{p}_{ij}^E(z_i, z_j) = \mathbf{p}_{ji}^E(z_j, z_i)$ by construction.

The node likelihood is

$$p_{\theta_X}(\hat{X}_1 \mid \hat{z}_{1:n}) = \prod_{i=1}^{n} \mathrm{Cat}\Big(\hat{X}_{1,i}; \mathbf{p}_i^X(\hat{z}_i)\Big), \tag{95}$$

and bond attributes are decoded only for edges present in the canonical target graph,

$$p_{\theta_E}(\hat{E}_1 \mid \hat{A}_1, \hat{z}_{1:n}) = \prod_{1 \le i < j \le n} \mathrm{Cat}\Big(\hat{E}_{1,ij}; \mathbf{p}_{ij}^E(\hat{z}_i, \hat{z}_j)\Big)^{\hat{A}_{1,ij}}. \tag{96}$$

The exponent $\hat{A}_{1,ij}$ masks non-edges, so bond-type reconstruction is evaluated only where the canonical target graph has an edge.

These likelihoods motivate the normalized Stage-1 reconstruction terms:

$$\mathcal{L}_X = \frac{1}{n}\sum_{i=1}^{n} \mathrm{CE}\Big(\hat{X}_{1,i}, f_X(\hat{z}_i)\Big), \qquad \mathcal{L}_E = \frac{1}{|\mathcal{E}_1|} \sum_{\substack{1 \le i < j \le n \\ \hat{A}_{1,ij}=1}} \mathrm{CE}\Big(\hat{E}_{1,ij}, \boldsymbol{\ell}_{ij}^E(\hat{z}_i, \hat{z}_j)\Big), \tag{97}$$

with $|\mathcal{E}_1| = \sum_{1 \le i < j \le n} \hat{A}_{1,ij}$

G.3 Design Rationale for the Latent-Graphon Decoder

This latent model is designed around two principles. First, edge existence is decoded only through the graphon. The edge decoder predicts edge type conditioned on the existence of an edge; it does not decide whether an edge should exist. This prevents the attribute decoder from competing with the graphon and preserves a clean probabilistic interpretation of the graphon as the Bernoulli adjacency decoder.

Second, the graphon is conditioned on the graph-level latent variable z_g, not on the node-level variables z_i. The graphon already depends on node position through the canonical lattice coordinates (s_i, s_j). Conditioning edge existence directly on node-level codes would reduce the model to a pairwise link-prediction decoder, eliminating the continuous graphon surface that is later used as a structured source distribution for flow matching (Section 3.2). By keeping edge existence graphon-based and using node latents only for local attributes, *GraphonFlow* preserves a clear separation between global topology and local semantics.

G.4 Conditional Velocity Field Architecture

The continuous evolution of the interpolated graph $G_t = (A_t, X_t, E_t)$ is parameterised by the conditional vector field $v_\theta(G_t, t, z_g, z_{1:n}) = \big(v_A, v_X, v_E\big)$, where $v_A \in \mathbb{R}^{n \times n}$, $v_X \in \mathbb{R}^{n \times d_x}$, and $v_E \in \mathbb{R}^{n \times n \times d_e}$ specify the rates of change of the adjacency, node-attribute, and edge-attribute states, respectively. The vector field is realised by an edge-aware graph transformer conditioned jointly on the integration time t, graph-level latent z_g, and node-level latents $z_{1:n}$. Its construction is permutation equivariant with respect to a simultaneous reordering of the node-indexed inputs, while the adjacency and edge outputs are constrained to remain symmetric for undirected graphs.

Let $h_i \in \mathbb{R}^{d_h}$ denote the hidden representation of node i, where d_h is the hidden feature dimension. At each layer, attention scores and pairwise messages are computed as

$$\text{score}_{ij} = \frac{q_i^\top k_j}{\sqrt{d_h}} + A_{t,ij}, \tag{98}$$

$$m_{ij} = \text{MLP}_{\text{msg}}(h_i, h_j, e_{ij}), \tag{99}$$

$$h_i' = h_i + \text{MLP}_{\text{node}}\left(\sum_j \psi_j(\text{score}_{ij}) m_{ij}\right), \tag{100}$$

$$e_{ij}' = e_{ij} + \text{MLP}_{\text{upd}}(m_{ij}), \tag{101}$$

where q_i and k_j are query and key projections of the node representations, and ψ_j denotes softmax normalisation over the neighbour index j.

The initial node representation incorporates both global and node-specific conditioning:

$$h_i^{(0)} = W_n X_i + \text{MLP}_t(t) + \text{MLP}_z(z_g) + \text{MLP}_{\text{node}}(z_i). \tag{102}$$

The time and graph-level latent terms provide conditioning shared across all nodes, whereas the final term injects node-specific information through z_i. Because the node-level latent variables transform together with their associated nodes under permutation, this conditioning preserves permutation equivariance while allowing node-dependent transport dynamics. Finally, the predicted structural velocities are symmetrised,

$$v_A \leftarrow \tfrac{1}{2}(v_A + v_A^\top), \qquad v_E \leftarrow \tfrac{1}{2}(v_E + v_E^\top),$$

ensuring that the continuous dynamics preserve the symmetry of undirected adjacency and edge-attribute tensors.

G.5 Flow Matching Loss Terms

Following Wijesinghe et al. (2026), the velocity field is trained by regressing the predicted velocities towards the rectified displacement between the structured source and canonical target graphs,

$$\mathcal{L}_{\text{vel}} = \frac{1}{n^2}\left\|v_A - (\hat{A}_1 - A_0)\right\|_F^2 + \frac{\lambda_x}{n\, d_x}\left\|v_X - (\hat{X}_1 - X_0)\right\|_F^2 + \frac{\lambda_e}{n^2 d_e}\left\|v_E - (\hat{E}_1 - E_0)\right\|_F^2, \tag{103}$$

where the normalisation factors correspond to Mean Squared Error (MSE) over the entries of the adjacency, node-feature, and edge-feature tensors, respectively.

To improve finite-step integration accuracy, we additionally supervise the predicted endpoint (Wijesinghe et al., 2026),

$$\tilde{A}_1 = A_t + (1-t)v_A, \qquad \tilde{X}_1 = X_t + (1-t)v_X, \qquad \tilde{E}_1 = E_t + (1-t)v_E, \tag{104}$$

through

$$\mathcal{L}_{\text{end}} = \frac{1}{n^2}\left\|\tilde{A}_1 - \hat{A}_1\right\|_F^2 + \frac{\lambda_x}{n\, d_x}\left\|\tilde{X}_1 - \hat{X}_1\right\|_F^2 + \frac{\lambda_e}{n^2 d_e}\left\|\tilde{E}_1 - \hat{E}_1\right\|_F^2. \tag{105}$$

Following Wijesinghe et al. (2026), chemical validity is encouraged using a differentiable soft valence penalty on the predicted endpoint,

$$\tilde{v}_i = \sum_j \tilde{A}_{1,ij} \sum_b \text{softmax}(\tilde{E}_{1,ij})_b \tilde{b}, \qquad \mathcal{L}_{\text{val}} = \frac{1}{n}\sum_{i=1}^{n} \max\left(0, \tilde{v}_i - v_{a_i}^{\max}\right), \tag{106}$$

where $\tilde{b}$ denotes the bond order associated with bond type b, and $v_{a_i}^{\max}$ is the maximum allowed valence for atom type a_i.

G.6 Derivation of the Connectivity Regularisers

Stage 1: graphon connectivity regulariser. The motivation for $\mathcal{L}_{\text{conn}}$ (Eq. 17) follows from classical random graph theory. An Erdős–Rényi graph $\mathbb{G}(n,p)$ undergoes a connectivity phase

transition when $p > \frac{1}{n}$, above which a giant connected component appears with high probability (Newman, 2002). Writing $\bar{W}_\psi(z_g) = \frac{1}{|\mathcal{S}|}\sum_{(i,j)\in\mathcal{S}} \bar{W}_\psi(s_i, s_j \mid z_g)$ for the mean of the *unsparsified* basis surface over a uniform lattice $\mathcal{S}$, the effective (sparsified) mean edge probability is $\bar{W}_n(z_g) = \bar{W}_\psi(z_g)\,\rho(z_g)$, and the corresponding connectivity threshold is $\rho(z_g) > \frac{1}{\bar{W}_\psi(z_g)n}$.

Because $\rho(z_g)$ is constant over the lattice (it depends only on z_g, not on lattice position), this identity holds exactly at the level of lattice averages as well: $\bar{W}_n(z_g) = \frac{1}{|\mathcal{S}|}\sum_{(i,j)\in\mathcal{S}} W_\psi(s_i, s_j \mid z_g)$, i.e. $\bar{W}_n(z_g)$ can equivalently be estimated by directly averaging the *sparsified* graphon over the lattice, without separately estimating $\bar{W}_\psi(z_g)$ and $\rho(z_g)$. We adopt this direct estimator for numerical convenience in Eq. 17.

Importantly, the sparsification factor is predicted as $\rho(z_g)$ rather than $\rho(n)$, because graph sparsity depends on the underlying graph family rather than graph size alone: graphs with the same number of nodes may exhibit substantially different densities and topologies, whereas the graph-level latent representation naturally captures these structural variations.

Stage 2: Lanczos estimation of algebraic connectivity. Since exact eigendecomposition of $\tilde{L}_1$ scales as $O(n^3)$, we estimate λ_2 using deflated Lanczos iteration, summarised in Algorithm 1 (Appendix D). Starting from a random vector deflated against the trivial eigenvector $\mathbf{u} = \mathbf{1}/\sqrt{n}$, we generate a k-dimensional Krylov subspace via the three-term Lanczos recurrence, re-deflating against $\mathbf{u}$ at every step to suppress numerical drift toward the trivial eigenvalue. Each step costs $O(n)$ for sparse graphs, for an overall time complexity of $O(kn)$. Only the two most recent Krylov vectors together with the accumulated scalar coefficients $\{\alpha_j, \beta_j\}_{j=1}^{k}$ are retained at any time, giving a memory footprint of $O(n+k)$ rather than the $O(kn)$ required by a fully reorthogonalised Lanczos scheme.

The accumulated coefficients define a $k \times k$ symmetric tridiagonal matrix $T = \mathrm{tridiag}(\beta_{1:k-1}, \alpha_{1:k}, \beta_{1:k-1})$, and λ_2 is estimated as the smallest eigenvalue of this tridiagonal projection, $\hat{\lambda}_2 = \lambda_{\min}(T)$, obtained via a dense eigendecomposition of T at cost $O(k^3)$, which is cheap for the small k used in practice. Because every α_j, β_j is a differentiable function of $\tilde{L}_1$ through the recurrence, and $\tilde{L}_1$ is itself a differentiable function of the predicted adjacency $\tilde{A}_1$, gradients propagate directly from $\hat{\lambda}_2$ back through T and $\tilde{L}_1$ to $\tilde{A}_1$, and consequently to the velocity network's parameters.

G.7 KL Divergence Definitions and Normalisation

The graph-level latent variable is regularised against the standard Gaussian prior,

$$\mathrm{KL}_g = \mathrm{KL}\big(q_\phi(z_g \mid G_1) \,\|\, p(z_g)\big) = \frac{1}{2d_g}\sum_{d=1}^{d_g}\left(\mu_{g,d}^2 + \sigma_{g,d}^2 - \log\sigma_{g,d}^2 - 1\right), \tag{107}$$

The node-level latent variables are regularised against the graph- and position-conditioned prior introduced in Eq. 6. Since this prior depends on z_g, we write the node-level KL explicitly as a function of the sampled graph-level latent:

$$\begin{aligned}\mathrm{KL}_n(z_g) &= \frac{1}{n}\sum_{i=1}^{n}\mathrm{KL}\big(q_\phi(\hat{z}_i \mid G_1) \,\|\, p_\omega(\hat{z}_i \mid z_g, s_i)\big) \\ &= \frac{1}{2\,n\,d_n}\sum_{i=1}^{n}\sum_{d=1}^{d_n}\left(\log\frac{\sigma_{\omega,i,d}^2}{\hat{\sigma}_{n,i,d}^2} + \frac{\hat{\sigma}_{n,i,d}^2 + (\hat{\mu}_{n,i,d} - \mu_{\omega,i,d})^2}{\sigma_{\omega,i,d}^2} - 1\right),\end{aligned} \tag{108}$$

The two terms are combined as

$$\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}_g + \nu\,\mathbb{E}_{q_\phi(z_g|G_1)}\left[\mathrm{KL}_n(z_g)\right], \tag{109}$$

where ν controls the relative contribution of the node-level KL regularisation.

The node-level KL depends on the graph-level latent through the graph- and position-conditioned prior $p(\hat{z}_i \mid z_g, s_i)$, and is therefore evaluated inside the expectation over $q_\phi(z_g \mid G_1)$ in the Stage 1

objective (Eq. 19). KL_g (Eq. 107) is averaged over the d_g graph-level latent dimensions, while $\mathrm{KL}_n(z_g)$ (Eq. 108) is averaged jointly over the n nodes and d_n node-level latent dimensions. This matches the normalisation convention used elsewhere in the Stage 1 objective: $\mathcal{L}_X$ is averaged over the n canonical nodes (Eq. 97), $\mathcal{L}_E$ over the $|\mathcal{E}_1|$ present canonical edges (Eq. 97), and $\mathcal{L}_{\text{graphon}}$ over all $\binom{n}{2}$ canonical node pairs (Eq. 11). Keeping every term on a comparable per-element scale prevents the objective in Eq. 19 from being dominated by whichever term has the largest raw cardinality (e.g. $\binom{n}{2}$ node pairs versus n nodes), which would otherwise require dataset-dependent retuning of β_g, β_{KL}, β_{ℓ_1}, and β_c across graph families of different typical size.

Monte Carlo estimation and KL annealing. During Stage 1, all stochastic reconstruction and regularisation terms are evaluated using the same reparameterised sample $(z_g, z_{1:n}) \sim q_\phi(\cdot \mid G_1)$. The expectation over the latent variables is approximated using one Monte Carlo sample per graph. To reduce posterior collapse, the KL weight $\beta_{\mathrm{KL}}(t)$ is linearly annealed during the first T_{warm} epochs.

G.8 Generation Procedure

Generation follows the probabilistic model learned during Stage-1 and the transport dynamics learned during Stage-2: *GraphonFlow* samples a latent graph representation, constructs a structured graph prior through the latent-conditioned graphon, and continuously refines this structured source into the final graph via K-step Euler integration of the learned velocity field, followed by a valence-aware projection to a discrete molecular graph. This differs from generating directly from independent noise, since the source graph is already structurally consistent with the learned graphon prior. The full procedure is given in Algorithm 3; its integration-error guarantee is Theorem 10.

Latent and source sampling. We first sample a graph-level latent variable $z_g \sim \mathcal{N}(0, I)$, and draw the graph size n from the empirical node-count distribution observed during training. Conditioned on z_g and the canonical position s_i, node-level latent variables are then sampled from the learned conditional prior $z_i \sim p_\omega(z_i \mid z_g, s_i)$, rather than from an independent standard Gaussian. This ensures that the sampling distribution used during generation is consistent with the latent distribution learned during training, avoiding the prior–posterior mismatch commonly encountered in latent-variable generative models.

Given z_g, the latent-conditioned graphon $W_\psi(\cdot, \cdot \mid z_g)$ defines a structured source graph by sampling the initial adjacency matrix $A_0 \sim \mathrm{Bernoulli}(W_\psi(s_i, s_j \mid z_g))$, where the node coordinates $\{s_i\}_{i=1}^n$ are placed on the canonical lattice introduced in Section 3.1. Initial node and edge attributes (X_0, E_0) are subsequently decoded from the sampled node-level latents $z_{1:n}$ using the frozen Stage-1 attribute decoders f_X, f_E (Eq. 95, Eq. 96), exactly as in Algorithm 3.

Integration and error propagation. Starting from this structured source graph, the conditional velocity field is integrated from $t = 0$ to $t = 1$ using K Euler steps (Algorithm 3), producing continuous predictions for adjacency, node attributes, and edge attributes. Under a Lipschitz assumption on v_θ (Assumption 2), Theorem 10 bounds the resulting global integration error in terms of the assumed per-step local truncation error, showing errors incurred early in transport are amplified by up to e^{L_v} through the remaining steps while errors at the final step are not amplified. Throughout transport, the model operates entirely in a continuous graph space, allowing gradients and structural information to evolve smoothly.

Valence-aware discretisation. We follow the valence-aware discretisation procedure of Wijesinghe et al. (2026): the predicted adjacency matrix is symmetrized and projected onto a binary graph by selecting the highest-scoring edges according to an expected bond budget, atom and bond types are recovered through categorical decoding, and a valence-aware pruning step removes bonds violating atom-specific valence constraints before the result is converted into a sanitized RDKit molecule. Learning and generation are therefore performed entirely in continuous graph space, with discrete molecular structures recovered only through this final projection step, applied exclusively at inference and not during training.

H Computational Complexity Analysis

Let B denote the minibatch size, n the number of nodes, m the number of observed edges, d_h the hidden dimension, L_{enc} the number of encoder layers, L the number of velocity-field layers, $|\mathcal{B}|$ the number of Fourier graphon basis functions, K_B the Bernstein polynomial degree, k the number of Lanczos iterations, and K the number of Euler integration steps used at generation.

Canonicalisation. Canonical labelling is performed using `pynauty`[1], a Python interface to NAUTY, during both stages of the training procedure (Algorithm 2). Its runtime is graph-dependent and does not admit a simple polynomial bound in n, since the underlying individualisation and refinement search depends on graph symmetries. In practice, NAUTY uses partition refinement and automorphism pruning to substantially reduce the search space, and its overhead is modest for the relatively small graphs considered here (Appendix E.1). Since canonicalisation is deterministic for each observed graph, the resulting permutations can in principle be precomputed and cached across training epochs.

Graphon evaluation and Stage-1 training. For the Fourier parameterisation, the current implementation explicitly evaluates $|\mathcal{B}|$ basis functions over all node pairs, producing an $n \times n \times |\mathcal{B}|$ basis tensor. Graphon evaluation therefore requires $O(n^2|\mathcal{B}|)$ time and $O(n^2|\mathcal{B}|)$ temporary basis storage. For the Bernstein parameterisation, let $r = K_B + 1$. Evaluating the one-dimensional Bernstein basis costs $O(nr)$, after which the graphon matrix is obtained according to Eq. 8. The two matrix products require $O(nr^2 + n^2 r)$ time, yielding $O(nK_B^2 + n^2 K_B)$ up to lower-order terms, without materialising an $n \times n \times |\mathcal{B}|$ tensor.

The graphon coefficient network depends only on z_g and hence has cost independent of n for fixed latent and hidden dimensions. The node decoder is linear in n, whereas the edge-attribute decoder is evaluated densely over all node pairs before reconstruction is restricted to observed edges, giving $O(m)$ scaling in graph size for fixed network width. In contrast, the L_{enc}-layer GINE encoder operates on the sparse observed edge list and scales linearly with $n + m$ for fixed hidden dimension. Consequently, the dense graphon and pairwise edge decoding operations dominate Stage 1 as n increases.

Stage-2 flow-matching training. *GraphonFlow* requires no pairwise graph matching or optimal-transport alignment between source and target graphs: canonical coordinates provide a deterministic correspondence, and each structured source G_0 is sampled directly from the frozen latent-conditioned graphon. The velocity field operates on dense interpolated tensors (A_t, X_t, E_t). In each velocity layer, dense attention requires $O(md_h)$ operations, while pairwise edge-message and edge-update require $O(md_h^2)$ operations. Thus the dominant cost of an L-layer velocity field is $O(Lmd_h^2)$ per graph.

The spectral connectivity loss constructs the soft Laplacian explicitly as a dense $n \times n$ matrix in the current implementation. Each Lanczos matrix–vector product therefore costs $O(n^2)$, giving $O(kn^2 + k^3)$ time for k iterations and the subsequent $k \times k$ tridiagonal eigendecomposition, with $O(n^2 + k)$ working memory. A sparse Laplacian implementation with $O(n)$ nonzeros would reduce the iterative component to $O(kn)$ time and $O(n + k)$ memory for sparse molecular graphs. Thus, Stage 2 training over E_2 epochs and M minibatches of B graphs scales as $O\big(E_2 MB\,(Lmd_h^2 + kn^2 + k^3)\big)$. Unlike alignment-based source construction, this cost is linear in the number of independently processed graphs and introduces no additional pairwise source–target matching step as in Flowette (Wijesinghe et al., 2026).

Sampling. Each sample is initialised by drawing $z_g \sim \mathcal{N}(0, I)$ and sampling a structured source graph from the learned latent-conditioned graphon. Generation then proceeds through K Euler integration steps, each requiring one forward pass of the conditional velocity field. The resulting dominant sampling complexity is $O\big(Km(Ld_h^2 + d_e)\big)$.

[1] pynauty: https://github.com/pdobsan/pynauty

I Extended Experimental Results

I.1 Synthetic Graph Generation

Hyperparameter Configurations. For all synthetic benchmarks, we optimise *GraphonFlow* using AdamW Loshchilov & Hutter (2019) with a learning rate of $1e^{-3}$, weight decay $1e^{-4}$, and batch size $B = 32$. We use hidden dimension $d_h = 128$, graph-level latent dimension $d_g = 64$, node-level latent dimension $d_n = 16$, and $L = 4$ velocity-field layers. In Stage 1, the graphon reconstruction, KL, coefficient regularisation, and connectivity weights are fixed to $\beta_g = 1.0$, $\beta_{\mathrm{KL}} = 0.5$, $\beta_{\ell_1} = 1e^{-3}$, and $\beta_c = 0.5$, respectively, with relative node-level KL weight $\nu = 1.0$. The KL coefficient is linearly annealed over the first $T_{\mathrm{warm}} = 10$ epochs. We use maximum frequency $M = 3$ for the Fourier graphon and polynomial degree $K_B = 6$ for the Bernstein graphon. For Stage 2, we set $\lambda_x = \lambda_e = 0.5$ and $\beta_{\mathrm{end}} = 1.0$ across all three datasets. Since the synthetic benchmarks contain no chemical constraints, we disable the valence objective by setting $\beta_{\mathrm{val}} = 0$. We use $\beta_{\mathrm{spec}} = 0.05$, 0.10, and 0.10 for Tree, SBM, and Ego-small, respectively, with corresponding spectral-connectivity margins $\gamma = 0.02$, 0.05, and 0.05. To account for differences in graph density, the minimum-density parameter in the Stage 1 connectivity regulariser is set to $\delta = 0.15$, while $\xi = 1e^{-4}$ is used for all three datasets. Generation is performed using 200 Euler integration steps for all three benchmarks.

Tree. Table 3 reports the results on the Tree benchmark. Both *GraphonFlow* variants generate exclusively valid, unique, and novel samples, yielding a V.U.N. score of 100%. They also obtain zero clustering and orbit discrepancies, indicating that the generated samples accurately reproduce these characteristic statistics of the target tree distribution. *GraphonFlow*–Fourier achieves a degree discrepancy of 0.0053, while *GraphonFlow*–Bernstein improves this to 0.0031. Although several baselines obtain smaller degree discrepancies, including DiGress and DeFoG at 0.0002 and Flowette at 0.0005, *GraphonFlow* simultaneously maintains 100% validity, uniqueness, novelty, and V.U.N. In particular, both parameterisations match Flowette on all four validity and diversity measures and on the clustering and orbit statistics.

SBM. The SBM results in Table 4 provide a more demanding test of whether the learned source distribution can represent community-structured graphs. Both *GraphonFlow* variants obtain zero orbit discrepancy and 100% uniqueness and novelty. *GraphonFlow*–Fourier achieves 98.5% validity and V.U.N., whereas the Bernstein parameterisation increases both measures to 98.9%. The Bernstein variant also improves the clustering discrepancy from 0.0400 to 0.0358 and the degree discrepancy from 0.0116 to 0.0098. Flowette attains 100% validity and V.U.N. and a lower degree discrepancy of 0.0076, while *GraphonFlow*–Bernstein produces a lower clustering discrepancy (0.0358 vs. 0.0508); both methods obtain zero orbit discrepancy. These results show that *GraphonFlow* preserves the principal structural characteristics of the SBM distribution while maintaining high validity and sample diversity.

Ego-small. Table 5 evaluates structural fidelity on Ego-small. Both *GraphonFlow* parameterisations achieve zero orbit discrepancy, matching Flowette and improving upon the other reported baselines on this metric. *GraphonFlow*–Fourier obtains degree and clustering discrepancies of 0.0260 and 0.0090, respectively, while *GraphonFlow*–Bernstein reduces these values to 0.0110 and 0.0075. The Bernstein parameterisation therefore improves over Flowette on both degree discrepancy (0.0110 versus 0.0435) and clustering discrepancy (0.0075 versus 0.0098), while retaining the same zero orbit discrepancy. EDGE remains stronger on degree and clustering statistics, with values of 0.0040 for both metrics.

Across the three synthetic benchmarks, both graphon parameterisations consistently recover important structural statistics of the target distributions. The Bernstein variant performs better than the Fourier variant on degree discrepancy across all three datasets and on clustering discrepancy for SBM and Ego-small, while the two parameterisations obtain identical orbit discrepancies. Together with the high validity and diversity observed on Tree and SBM, these results indicate that the learned latent-conditioned graphon provides an effective structured source for continuous graph transport across distinct synthetic graph families.

Table 3: Graph generation results on the synthetic Tree dataset.

Model	Deg. ↓	Clus. ↓	Orbit ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N ↑
GRAN (Liao et al., 2019)	0.1884	0.0080	0.0199	0.0	**100.0**	**100.0**	0.0
GraphGen (Goyal et al., 2020)	0.0105	**0.0000**	**0.0000**	95.0	**100.0**	**100.0**	95.0
BiGG (Dai et al., 2020)	0.0014	**0.0000**	**0.0000**	100	87.5	50.0	75.0
DiGress (Vignac et al., 2023)	0.0002	**0.0000**	**0.0000**	90.0	**100.0**	**100.0**	90.0
HSpectre (Bergmeister et al., 2024)	0.0004	**0.0000**	**0.0000**	82.5	**100.0**	**100.0**	82.5
BwR (Diamant et al., 2023)	0.0016	0.1239	0.0003	0.0	**100.0**	**100.0**	0.0
EDGE (Chen et al., 2023)	0.2678	**0.0000**	0.7357	0.0	7.5	**100.0**	0.0
G2PT (Chen et al., 2025)	0.0020	**0.0000**	**0.0000**	-	-	-	99.0
DeFoG (Qin et al., 2025)	**0.0002**	**0.0000**	**0.0000**	96.5	**100.0**	**100.0**	96.5
Flowette (Wijesinghe et al., 2026)	0.0005 ± 0.002	**0.0000** ± 0.0	**0.0000** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0
GraphonFlow(Fourier)	0.0053 ± 0.001	**0.0000** ± 0.0	**0.0000** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0
GraphonFlow(Bernstein)	0.0031 ± 0.004	**0.0000** ± 0.0	**0.0000** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0

Table 4: Graph generation results on the synthetic SBM dataset.

Model	Deg. ↓	Clus. ↓	Orbit ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N ↑
DiGress (Vignac et al., 2023)	0.0018	0.0485	0.0415	60.0	**100.0**	**100.0**	60.0
HSpectre (Bergmeister et al., 2024)	0.0141	0.0528	0.0809	75.0	**100.0**	**100.0**	75.0
BwR (Diamant et al., 2023)	0.0478	0.0638	0.1139	7.5	**100.0**	**100.0**	7.5
Cometh (Siraudin et al., 2024)	0.0020	0.0498	0.0383	75.0	**100.0**	**100.0**	75.0
GruM (Jo et al., 2024)	0.0015	0.0589	0.0450	-	-	-	85.0
DisCo (Xu et al., 2024)	0.0006	0.0266	0.0510	66.2	**100.0**	**100.0**	66.2
CatFlow (Eijkelboom et al., 2024)	0.0012	0.0498	0.0357	-	-	-	85.0
G2PT (Chen et al., 2025)	0.0035	**0.0120**	0.0007	-	-	-	**100.0**
DeFoG (Qin et al., 2025)	**0.0006**	0.0517	0.0556	90.0	90.0	90.0	90.0
Flowette (Wijesinghe et al., 2026)	0.0076 ± 0.0035	0.0508 ± 0.01	**0.0000** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0
GraphonFlow(Fourier)	0.0116 ± 0.007	0.0400 ± 0.05	**0.0000** ± 0.0	98.5 ± 0.08	**100.0** ± 0.0	**100.0** ± 0.0	98.5 ± 0.05
GraphonFlow(Bernstein)	0.0098 ± 0.005	0.0358 ± 0.03	**0.0000** ± 0.0	98.9 ± 0.05	**100.0** ± 0.0	**100.0** ± 0.0	98.9 ± 0.03

I.2 Molecular Graph Generation

We use the same evaluation metrics for molecular graph generation as used in Flowette (Wijesinghe et al., 2026). For molecular generation, we benchmark *GraphonFlow* against a broad set of established graph generative models. On Guacamol and MOSES, we compare with DiGress (Vignac et al., 2023), DisCo Xu et al. (2024), Cometh (Siraudin et al., 2024), GraphBFN Song et al. (2025), G2PT (Chen et al., 2025), GEEL Jang et al. (2024), DeFoG (Qin et al., 2025), and Flowette (Wijesinghe et al., 2026). For QM9 and ZINC250K, the comparison comprises EDP-GNN (Niu et al., 2020), GraphARM (Kong et al., 2023), DiGress (Vignac et al., 2023), GDSS and GDSS+TF (Jo et al., 2022), GruM (Jo et al., 2024), SID (Boget, 2025), DeFoG (Qin et al., 2025), GraphBFN (Song et al., 2025), GBD (Liu et al., 2025), and Flowette (Wijesinghe et al., 2026).

Hyperparameter Configurations. For all molecular benchmarks, *GraphonFlow* is optimised using AdamW (Loshchilov & Hutter, 2019) with a learning rate of $1e^{-3}$, weight decay $1e^{-4}$, and batch size $B = 32$. We use hidden dimension $d_h = 128$, graph-level latent dimension $d_g = 64$, and node-level latent dimension $d_n = 16$. Both stages are trained for 200 epochs. Unless otherwise stated, the same configuration is used for QM9, ZINC250K, GuacaMol, and MOSES. For Stage 1, corresponding to the objective in Eq. 19, we set the graphon reconstruction weight to $\beta_g = 1.0$, the maximum KL weight to $\beta_{\mathrm{KL}} = 0.5$, the relative node-level KL weight to $\nu = 1.0$, the coefficient regularisation weight to $\beta_{\ell_1} = 1e^{-3}$, and the graphon connectivity weight to $\beta_c = 0.5$. The KL coefficient is linearly annealed from zero to β_{KL} over the first $T_{\mathrm{warm}} = 10$ epochs. For the connectivity objective in Eq. 17, we use $\xi = 1e^{-4}$ and $\delta = 0.15$. For Stage 2 (Eq. 20), we use $\lambda_x = \lambda_e = 0.5$ for the node- and edge-attribute components of both $\mathcal{L}_{\mathrm{vel}}$ and $\mathcal{L}_{\mathrm{end}}$. The auxiliary objective weights are set to $\beta_{\mathrm{end}} = 1.0$, $\beta_{\mathrm{val}} = 0.5$, and $\beta_{\mathrm{spec}} = 0.1$. For the spectral-connectivity objective in Eq. 18, the algebraic-connectivity margin is $\gamma = 0.05$. For the latent-conditioned graphon, the Fourier parameterisation uses maximum frequency $M = 6$, whereas the Bernstein parameterisation uses polynomial degree $K_B = 6$. At generation time, graph size is sampled from the empirical node-count distribution and the learned conditional flow is integrated using 200 Euler steps.

QM9 and ZINC250K. As shown in Table 1, both *GraphonFlow* variants achieve strong molecular validity while maintaining high sample diversity and close agreement with the structural distribution of the training data. On QM9, *GraphonFlow*–Fourier reaches 99.90% validity and 99.40% uniqueness, while the Bernstein parameterisation further improves these values to 99.93% and 99.65%, respectively. The Bernstein variant also obtains an NSPDK of 0.0002, matching the strongest re-

Table 5: Graph generation results on the synthetic Ego-small dataset.

Model	Deg. ↓	Clus. ↓	Orbit ↓
GraphRNN (You et al., 2018)	0.0768	1.1456	0.1087
GRAN (Liao et al., 2019)	0.5778	0.3360	0.0406
GraphCNF (Lippe & Gavves, 2021)	0.1010	0.7654	0.0820
DiGress (Vignac et al., 2023)	0.0708	0.0092	0.1205
DiscDDPM (Haefeli et al., 2022)	0.4613	0.1681	0.0633
GDSS (Jo et al., 2022)	0.8189	0.6032	0.3315
EDGE (Chen et al., 2023)	**0.0040**	**0.0040**	0.0008
Flowette (Wijesinghe et al., 2026)	0.0435 $\pm$ 0.004	0.0098 $\pm$ 0.0025	**0.0000** $\pm$ 0.0
GraphonFlow(Fourier)	0.0260 $\pm$ 0.006	0.0090 $\pm$ 0.004	**0.0000** $\pm$ 0.0
GraphonFlow(Bernstein)	0.0110 $\pm$ 0.003	0.0075 $\pm$ 0.005	**0.0000** $\pm$ 0.0

sults among the compared methods up to the best reported value of 0.0001 by SID. These results indicate that the learned graphon source can preserve the characteristic molecular structure of QM9 without sacrificing the diversity of generated samples. The same behaviour extends to the more challenging ZINC250K benchmark. *GraphonFlow*–Bernstein achieves 99.93% validity, 100.00% uniqueness, and an NSPDK of 0.0004. In comparison, Flowette obtains 99.90% validity, 100.00% uniqueness, and an NSPDK of 0.0006. The Fourier variant remains competitive, reaching 99.86% validity, 100.00% uniqueness, and 0.0006 NSPDK. The consistently low NSPDK values are particularly relevant to *GraphonFlow*, since they indicate that learning the source connectivity distribution through a latent-conditioned graphon does not merely improve chemical feasibility, but also preserves important structural statistics of the molecular graph distribution.

Guacamol and MOSES. Table 2 evaluates whether these gains persist on larger and more diverse molecular collections. On Guacamol, *GraphonFlow*–Bernstein obtains 99.30% validity and 99.00% V.U.N., while *GraphonFlow*–Fourier achieves 99.05% validity and 98.50% V.U.N. Both improve upon the corresponding Flowette values of 98.6% and 98.0%. For distributional agreement, the Bernstein variant achieves a KL-divergence score of 97.7, matching DeFoG and remaining close to Flowette's 97.9. This suggests that the improvement in valid and non-redundant generation is obtained while retaining close agreement with the benchmark molecular distribution. On MOSES, both *GraphonFlow* variants achieve 100.0% uniqueness. *GraphonFlow*–Bernstein further obtains 98.85% novelty, 0.59 SNN, and 16.0 scaffold similarity, compared with 98.10%, 0.58, and 15.3, respectively, for Flowette. The Fourier parameterisation also performs strongly, with 98.45% novelty and a scaffold score of 15.7. Notably, the Bernstein model matches the highest SNN value reported in the table and the highest scaffold score, while preserving the highest reported uniqueness and novelty. Together, these results show that *GraphonFlow* generates diverse molecules while retaining substantial similarity to the reference molecular and scaffold distributions.

Across the four molecular benchmarks, the Bernstein graphon provides the most consistent *GraphonFlow* results, particularly on validity, novelty, and structural-distribution metrics. This trend suggests that its smooth polynomial parameterisation is well suited to representing the latent connectivity patterns required for molecular generation. More broadly, the results support the central design of *GraphonFlow*: learning a structured, latent-dependent source distribution can provide the flow with graph states that already reflect important properties of the target domain, while the subsequent continuous transport refines these states into chemically valid molecular graphs.

I.3 Runtime Analysis

In Table 6, we compare the empirical runtime of *GraphonFlow* with DeFoG (Qin et al., 2025) and Flowette (Wijesinghe et al., 2026) under a matched computational setting, using the same batch size (B=512), total epoch budget, and four benchmarks considered in Flowette's runtime analysis (Wijesinghe et al., 2026). For *GraphonFlow*, the reported training time includes both Stage 1 probabilistic structural learning and Stage 2 flow-matching optimisation. The results are summarised in Table 6.

Training time. *GraphonFlow* consistently requires less total training time than Flowette across the four benchmarks. The main computational distinction arises from how source and target structures are brought into correspondence during training. Flowette constructs structure-preserving training pairs using FGW-based graph comparison followed by assignment (Wijesinghe et al., 2026), introducing

Table 6: Runtime comparison of DeFoG, Flowette, and *GraphonFlow*, under a matched batch size ($B = 512$) and total epoch budget. *GraphonFlow*'s training time is reported per stage (Stage 1: VAE + graphon; Stage 2: flow matching) and as a total; Flowette's training time is reported as model training + FGW/Hungarian pairing overhead.

Dataset	DeFoG Train (h)	Flowette Train (h)	GraphonFlow Train (h)	DeFoG Sample (h)	Flowette Sample (h)	GraphonFlow Sample (h)	Graphs Sampled
Tree	8.00	8.52	8.10	0.07	0.04	0.05	40
QM9	6.50	7.46	7.00	0.20	0.16	0.14	10000
ZINC250K	14.00	15.35	14.70	4.80	3.70	3.75	10000
MOSES	46.00	48.18	46.80	5.00	4.30	4.10	25000

an additional graph-matching overhead on top of velocity-field optimisation. *GraphonFlow* removes this alignment step: canonical graph labelling establishes a deterministic node coordinate system, while the learned latent-conditioned graphon directly provides the structured source distribution used during Stage 2. Consequently, no FGW or Hungarian matching is required between source and target graphs. Although *GraphonFlow* incurs the additional Stage 1 cost of learning the VAE and latent graphon, this cost is offset by eliminating the explicit alignment machinery used by Flowette. Empirically, total training time decreases from 8.52 to 8.10 hours on Tree, 7.46 to 7.00 hours on QM9, 15.35 to 14.70 hours on ZINC250K, and 48.18 to 46.80 hours on MOSES.

Sampling time. *GraphonFlow* and Flowette exhibit comparable sampling efficiency. Once training is complete, neither method performs graph matching during generation; both generate samples by numerically integrating a learned velocity field for K Euler steps. Accordingly, their dominant generation cost is determined by repeated forward evaluations of the velocity network rather than the alignment procedure used during training. This similarity is reflected empirically: *GraphonFlow* requires 0.05, 0.14, 3.75, and 4.10 hours on Tree, QM9, ZINC250K, and MOSES, respectively, compared with 0.04, 0.16, 3.70, and 4.30 hours for Flowette. Thus, neither method uniformly dominates the other in sampling time: Flowette is slightly faster on Tree and ZINC250K, whereas *GraphonFlow* is slightly faster on QM9 and MOSES. Both flow-matching approaches are substantially faster than DeFoG on the larger molecular benchmarks, most notably ZINC250K and MOSES.

I.4 Visualising the Learned Molecular Representation

To qualitatively examine the structure captured by *GraphonFlow*, we visualise its learned molecular representation using Locally Linear Embedding (LLE) (Roweis & Saul, 2000). LLE constructs a low-dimensional embedding while approximately preserving local neighbourhood relationships, providing a qualitative view of how molecules are organised in the learned representation. Figure 6 shows two-dimensional LLE embeddings for QM9 and GuacaMol, with representative molecular structures displayed next to selected points across each embedding.

The visualisations in Fig. 6 reveal organised yet dataset-dependent molecular spaces. For QM9, the embedding is relatively compact, with representative molecules spanning different chain and ring structures across the two-dimensional space. In contrast, GuacaMol exhibits a broader and more heterogeneous embedding, consistent with its greater structural diversity, with representative molecules ranging from extended chains to larger cyclic and more complex structures. These qualitative patterns suggest that the learned representation captures meaningful variation across molecular graph structures rather than collapsing molecules to a single region. We emphasise that the LLE projections are used here as a qualitative visualisation of the learned representation; the quantitative quality of generated molecular distributions is evaluated using the generation metrics reported in Section 5.1.

I.5 Ablation Study

I.5.1 Ablation of Graphon Prior Contribution

We isolate the contribution of the structured source prior by modifying only the mechanism used to generate the source adjacency A_0, while keeping the latent encoder, node-level latent prior, attribute decoders, conditional velocity field, Stage 2 objectives, training budget, and valence-

Table 7: Ablation of the source topology prior on QM9 and ZINC250K. The density-matched ER prior controls only graph-size-dependent expected density; the unconditional graphon additionally learns shared position-dependent connectivity structure; and the proposed latent-conditioned graphon further conditions this structure on z_g. All other components are kept fixed across variants.

	QM9				ZINC250K			
Variant	Valid↑	Unique↑	Novel↑	NSPDK↓	Valid↑	Unique↑	Novel↑	NSPDK↓
ER	88.65 ± 0.03	87.50 ± 0.02	88.00 ± 0.05	0.0097 ± 0.0005	89.05 ± 0.01	88.65 ± 0.04	88.75 ± 0.06	0.0084 ± 0.0003
UG (Fourier)	93.50 ± 0.02	94.65 ± 0.03	94.85 ± 0.05	0.0045 ± 0.0002	94.95 ± 0.04	95.20 ± 0.01	95.00 ± 0.05	0.0030 ± 0.0001
UG (Bernstein)	94.25 ± 0.03	94.90 ± 0.06	95.35 ± 0.05	0.0025 ± 0.0003	95.05 ± 0.01	95.50 ± 0.03	95.60 ± 0.06	0.0023 ± 0.0003
LCG (Fourier)	**99.90** ± 0.05	**99.40** ± 0.06	**99.10** ± 0.04	**0.0003** ± 0.0004	**99.86** ± 0.02	**100.00** ± 0.0	**100.00** ± 0.0	**0.0006** ± 0.0006
LCG (Bernstein)	**99.93** ± 0.03	**99.65** ± 0.02	**99.25** ± 0.05	**0.0002** ± 0.0005	**99.93** ± 0.04	**100.00** ± 0.0	**100.00** ± 0.0	**0.0004** ± 0.0003

aware discretisation (Appendix G.8) unchanged. We consider three increasingly expressive source distributions: (i) a density-matched Erdős–Rényi (ER) prior, which controls only the expected graph density; (ii) an unconditional learned graphon $W_\psi(s_i, s_j)$, which captures position-dependent structural regularities shared across the dataset (UG); and (iii) the proposed latent-conditioned graphon $W_\psi(s_i, s_j \mid z_g)$, whose connectivity law additionally varies with the graph-level latent representation (LCG). This design separately tests the benefit of learning a structured graphon source and the additional contribution of conditioning that source on z_g.

Across all variants, the node-level latent prior $p_\omega(z_i \mid z_g, s_i)$ and attribute decoders (f_X, f_E) are retained unchanged, such that source node and edge attributes (X_0, E_0) continue to be generated from $z_{1:n}$. Thus, the ablation changes the source topology A_0 without simultaneously changing its attribute-generation mechanism.

For the ER control, we match graph density as a function of graph size rather than selecting a single global edge probability. Specifically, for graphs with n nodes, we estimate $p_n = \mathbb{E}_{G\sim\mathcal{D}:\,|V(G)|=n}\left[\frac{2|E(G)|}{n(n-1)}\right]$, and sample $A_{0,ij} \overset{\text{ind}}{\sim} \text{Bernoulli}(p_n)$. Hence, the ER baseline approximately preserves the expected sparsity at each graph size but contains no position-dependent or latent-dependent connectivity structure. The density-matched ER control has no basis parameterization and is therefore reported as a single row; the unconditional and latent-conditioned graphons are evaluated under both basis families (Section 3.1).

In this setting, three priors represent a natural hierarchy of source-topology expressiveness. A density-matched ER model specifies only the expected edge density and is recovered as a constant graphon. An unconditional graphon generalises this model by allowing connectivity probabilities to vary over the canonical coordinates (s_i, s_j), thereby representing shared structural heterogeneity across the data distribution. The latent-conditioned graphon further generalises the unconditional model by allowing its connectivity law to vary with z_g; removing this dependence recovers the unconditional case. Thus, the three priors satisfy, $\mathcal{P}_{\text{ER}} \subseteq \mathcal{P}_{\text{uncond}} \subseteq \mathcal{P}_{\text{cond}}$, at the level of representable source distributions, while the results in Table 7 show how increasing the structural expressiveness of the source prior affects generation performance.

I.5.2 Ablation of Canonical Coordinates Conditioning

We study the role of canonical structural coordinates in learning the latent graphon prior. All variants use the same latent-conditioned graphon architecture $W_\psi(s_i, s_j \mid z_g)$ (Section 3.1) and flow-matching objective (Section 3.2); only the node ordering used for coordinate assignment and, where noted, the conditioning of the node-level prior on s_i, are changed. Specifically, we compare four variants: (i) *random permutation*, where a fresh independent random node permutation is drawn for each graph at every training step and applied jointly and consistently to the adjacency, node attributes, and edge attributes (Eq. 83), so the correspondence between graph structure and graphon coordinates changes throughout training; (ii) *no canonicalisation*, where graphs retain their observed node ordering; (iii) *canonical without position conditioning*, where graphs are canonicalised but the node prior is restricted to $p_\omega(z_i \mid z_g)$; and (iv) the full model, which combines canonicalisation $C(G)$ with the fixed canonical coordinates s_i and the position-conditioned node prior $p_\omega(z_i \mid z_g, s_i)$. This ablation separates the contribution of resolving permutation ambiguity from the additional benefit of conditioning node-level latent structure on canonical position.

Table 8: Ablation study on canonical structural coordinates for QM9 and ZINC250K. We compare training without canonicalisation, random node permutation resampled at every training step, canonicalisation without position conditioning of the node prior, and the full model using canonicalisation together with $p_\omega(z_i \mid z_g, s_i)$. All variants share the same latent-conditioned graphon architecture and flow-matching objective; only node-coordinate assignment and node-prior conditioning are changed.

	QM9				ZINC250K			
Variant	Valid↑	Unique↑	Novel↑	NSPDK↓	Valid↑	Unique↑	Novel↑	NSPDK↓
RandPerm (Fourier)	88.35 ± 0.05	88.77 ± 0.04	88.60 ± 0.03	0.0135 ± 0.0005	88.57 ± 0.01	88.99 ± 0.05	88.78 ± 0.08	0.0110 ± 0.0006
RandPerm (Bernstein)	88.67 ± 0.06	88.95 ± 0.03	89.00 ± 0.04	0.0115 ± 0.0003	88.89 ± 0.04	89.14 ± 0.07	89.44 ± 0.04	0.0105 ± 0.0005
NoCanon (Fourier)	92.75 ± 0.04	93.00 ± 0.05	93.40 ± 0.06	0.0095 ± 0.0005	92.90 ± 0.05	93.35 ± 0.05	93.95 ± 0.03	0.0075 ± 0.0004
NoCanon (Bernstein)	93.10 ± 0.03	93.60 ± 0.04	93.70 ± 0.03	0.0065 ± 0.0006	93.45 ± 0.01	93.80 ± 0.04	94.00 ± 0.03	0.0050 ± 0.0007
CanonNoPos (Fourier)	95.90 ± 0.01	95.45 ± 0.03	95.00 ± 0.06	0.0014 ± 0.0003	96.00 ± 0.05	95.80 ± 0.07	96.10 ± 0.04	0.0010 ± 0.0005
CanonNoPos (Bernstein)	96.20 ± 0.03	95.80 ± 0.01	95.50 ± 0.05	0.0018 ± 0.0005	96.50 ± 0.04	96.00 ± 0.03	96.25 ± 0.02	0.0015 ± 0.0004
Canon+Pos (Full model) (Fourier)	**99.90** ± 0.05	**99.40** ± 0.06	**99.10** ± 0.04	**0.0003** ± 0.0004	**99.86** ± 0.02	**100.00** ± 0.0	**100.0** ± 0.0	**0.0006** ± 0.0006
Canon+Pos (Full model) (Bernstein)	**99.93** ± 0.03	**99.65** ± 0.02	**99.25** ± 0.05	**0.0002** ± 0.0005	**99.93** ± 0.04	**100.00** ± 0.0	**100.00** ± 0.0	**0.0004** ± 0.0003

Without canonicalisation, the same graphon coordinate pair may receive inconsistent edge supervision across isomorphic but differently labelled graphs. Random permutation further exposes the model to this ambiguity by changing the correspondence between graph structure and graphon coordinates across training instances. Canonicalisation instead establishes a deterministic correspondence between graph structure and the canonical lattice. Comparing the final two variants then isolates whether explicitly conditioning the node prior on s_i provides additional benefit once this correspondence has been established. The no-canonicalisation and random-permutation variants provide empirical evidence for the ordering-dependence that Theorem 3 shows is otherwise eliminated by canonical supervision.

To evaluate the role of canonical structural coordinates in the *GraphonFlow* training objective, we conduct an ablation study by selectively removing its key elements as shown in Table 8.

- **RandPerm**: This variant replaces canonicalisation with a fresh independent random node permutation, drawn for each graph at every training step, so the correspondence between graph structure and graphon coordinates changes throughout training.
- **NoCanon**: This variant removes canonicalisation entirely; graphs retain their observed node ordering, and the graphon is supervised directly against A_1 rather than $\hat{A}_1$.
- **CanonNoPos**: This variant canonicalises graphs via NAUTY but restricts the node-level prior to $p_\omega(z_i \mid z_g)$, removing conditioning on the canonical position s_i.
- **Canon+Pos (Full model)**: This variant combines canonicalisation with the position-conditioned node prior $p_\omega(z_i \mid z_g, s_i)$, matching the full *GraphonFlow* model.

Table 8 shows a consistent improvement as the correspondence between graph structure and canonical coordinates is progressively strengthened. RandPerm performs worst across both datasets, consistent with the fact that repeatedly changing the structure-to-coordinate correspondence provides inconsistent supervision to the coordinate-dependent graphon. Retaining the observed ordering (NoCanon) improves all metrics, but remains substantially below CanonNoPos, showing that a deterministic canonical correspondence provides a more effective structural reference for learning the graphon prior. Finally, Canon+Pos yields the strongest overall performance, indicating that canonicalisation alone does not account for the full gain: explicitly conditioning the node-level prior on s_i provides additional information once a stable canonical coordinate system has been established. The same progression is observed for both Fourier and Bernstein parameterisations and on both QM9 and ZINC250K, supporting the role of canonical structural coordinates across both graphon parameterisations considered.

I.5.3 Ablation of Two-Stage vs. Joint Optimisation

GraphonFlow trains the graphon prior and the velocity field in two separate stages, freezing (ϕ, ψ, ω) before optimising θ (Sections 4.2, 4.3). This separation is motivated by avoiding the non-stationary source distribution induced by joint optimisation. In particular, the graphon must first learn the target structural distribution before the flow is trained to refine samples drawn from it. If W_ψ is still evolving, the source graphs $A_0 \sim \mathrm{Bernoulli}(W_\psi(\cdot, \cdot \mid z_g))$ and, consequently, the interpolation paths presented to the velocity field also change throughout training. We isolate the contribution of this

Table 9: Ablation study of two-stage vs. joint optimisation on QM9 and ZINC250K, under a matched total gradient-step budget across variants.

	QM9				ZINC250K			
Variant	Valid↑	Unique↑	Novel↑	NSPDK↓	Valid↑	Unique↑	Novel↑	NSPDK↓
Joint (Fourier)	95.37 ± 0.05	94.38 ± 0.03	95.00 ± 0.07	0.0025 ± 0.0005	96.10 ± 0.02	95.45 ± 0.06	95.60 ± 0.04	0.0020 ± 0.0003
Joint (Bernstein)	95.90 ± 0.02	94.85 ± 0.01	95.61 ± 0.04	0.0016 ± 0.0003	96.50 ± 0.04	95.70 ± 0.02	96.00 ± 0.05	0.0013 ± 0.0005
TwoStage (Fourier)	**99.90** ± 0.05	**99.40** ± 0.06	**99.10** ± 0.04	**0.0003** ± 0.0004	**99.86** ± 0.02	**100.00** ± 0.0	**100.0** ± 0.0	**0.0006** ± 0.0006
TwoStage (Bernstein)	**99.93** ± 0.03	**99.65** ± 0.02	**99.25** ± 0.05	**0.0002** ± 0.0005	**99.93** ± 0.04	**100.00** ± 0.0	**100.00** ± 0.0	**0.0004** ± 0.0003

design choice by comparing the proposed staged, frozen schedule against fully joint optimisation of all parameters. Results are reported in Table 9.

- **Joint**: All parameters $(\phi, \psi, \omega, \theta)$ are optimised jointly from initialisation, minimising $\mathcal{L}_{S1} + \mathcal{L}_{S2}$ at every step; A_0 is sampled from the still-training graphon at each iteration.
- **TwoStage (Ours)**: The proposed schedule: Stage 1 is optimised to convergence, (ϕ, ψ, ω) are then frozen, and only θ is optimised on $\mathcal{L}_{S2}$ during Stage 2.

Table 9 shows a clear and consistent advantage for the proposed two-stage optimisation. On QM9, replacing joint training with the staged schedule increases validity from 95.37% to 99.90% for the Fourier graphon and from 95.90% to 99.93% for the Bernstein graphon. Similar improvements are observed for uniqueness and novelty, while NSPDK decreases from 0.0025 to 0.0003 for Fourier and from 0.0016 to 0.0002 for Bernstein. The effect is equally pronounced on ZINC250K: the two-stage variants reach approximately 99.9% validity and 100% uniqueness and novelty, whereas joint optimisation remains below 97% validity and 96% uniqueness. Structural fidelity also improves substantially, with NSPDK decreasing from 0.0020 to 0.0006 for Fourier and from 0.0013 to 0.0004 for Bernstein.

These gains are consistent with the motivation for separating the two learning problems in *Graphon-Flow*. Under joint optimisation, the source distribution $A_0 \sim \mathrm{Bernoulli}(W_\psi(\cdot, \cdot \mid z_g))$ continues to change as ψ is updated. Consequently, the conditional flow is trained against a non-stationary source distribution: changes to the graphon alter both the initial graph structures and the transport paths $G_t = (1-t)G_0 + t\hat{G}_1$ on which the velocity field is supervised. The velocity field must therefore adapt simultaneously to the desired target transport and to changes in its own source distribution. In the proposed two-stage procedure, Stage 1 first learns the latent representation and structured graphon prior; freezing (ϕ, ψ, ω) thereafter fixes the source-generating mechanism throughout Stage 2. This provides θ with a stable distribution of structured source graphs and a consistent family of interpolation paths on which to learn the conditional vector field.

I.5.4 Ablation Analysis of Stage-1 Regularization Terms

The Stage-1 objective $\mathcal{L}_{S1}$ (Eq. 19) combines the reconstruction terms $\mathcal{L}_X + \mathcal{L}_E + \beta_g \mathcal{L}_{\text{graphon}}$, which define the VAE's likelihood and are therefore never removed, with three regularizers governing distinct aspects of the latent representation and the graphon prior: $\beta_{\text{KL}}(t)\mathcal{L}_{\text{KL}}$ regularizes the latent posteriors toward their (graph- and position-conditioned) priors, which is what makes the prior sampling $z_g \sim \mathcal{N}(0, I)$ used at generation (App. G.8) consistent with the training-time posterior; $\beta_{\ell_1}\mathcal{L}_{\ell_1}$ encourages compact graphon basis coefficients, discouraging over-parameterized or noisy graphon surfaces; and $\beta_c \mathcal{L}_{\text{conn}}(z_g, n)$ (Section 3.3) encourages the sampled source graph to be sufficiently connected before Stage-2 transport begins.

To evaluate the role of each Stage-1 regularizer, we conduct an ablation study by selectively removing its key elements as shown in Table 10. Since the source-topology and canonical-coordinate ablations exhibit consistent qualitative trends across both graphon basis families, we report the remaining regularisation ablations using the Bernstein basis only.

- **NoReg (only $\mathcal{L}_X + \mathcal{L}_E + \beta_g \mathcal{L}_{\text{graphon}}$)**: This variant removes all three regularization terms and keeps only the reconstruction objective.
- **NoKLReg (no $\mathcal{L}_{\text{KL}}$)**: This variant removes only the regularization term $\mathcal{L}_{\text{KL}}$.
- **NoL1Reg (no $\mathcal{L}_{\ell_1}$)**: This variant removes only the regularization term $\mathcal{L}_{\ell_1}$.

Table 10: Ablation study of Stage-1 regularization terms on QM9 and ZINC250K. Recall from Eq. 19, the components of the Stage-1 objective: the reconstruction terms $\mathcal{L}_X + \mathcal{L}_E + \beta_g \mathcal{L}_{\text{graphon}}$, the KL regularizer $\mathcal{L}_{\text{KL}}$, the graphon sparsity regularizer $\mathcal{L}_{\ell_1}$, and the connectivity regularizer $\mathcal{L}_{\text{conn}}$.

	QM9				ZINC250K			
Variant (Bernstein)	Valid↑	Unique↑	Novel↑	NSPDK↓	Valid↑	Unique↑	Novel↑	NSPDK↓
NoReg	83.35 ± 0.01	85.46 ± 0.05	85.70 ± 0.02	0.0406 ± 0.0005	87.86 ± 0.03	87.05 ± 0.03	88.56 ± 0.04	0.0487 ± 0.0003
NoKLReg	87.15 ± 0.01	88.58 ± 0.03	89.00 ± 0.03	0.0195 ± 0.0004	89.98 ± 0.04	90.65 ± 0.06	92.57 ± 0.05	0.0205 ± 0.0001
NoL1Reg	95.37 ± 0.01	97.35 ± 0.03	96.50 ± 0.04	0.0010 ± 0.0003	96.45 ± 0.02	97.80 ± 0.04	97.86 ± 0.05	0.0020 ± 0.0001
NoConnReg	92.32 ± 0.03	95.25 ± 0.04	95.00 ± 0.05	0.0025 ± 0.0003	94.05 ± 0.01	96.58 ± 0.05	96.80 ± 0.03	0.0036 ± 0.0002
NoStructReg	90.25 ± 0.05	92.27 ± 0.06	93.00 ± 0.03	0.0125 ± 0.0005	91.65 ± 0.03	94.05 ± 0.04	93.25 ± 0.03	0.0157 ± 0.0003
Full model	**99.93** ± 0.03	**99.65** ± 0.02	**99.25** ± 0.05	**0.0002** ± 0.0005	**99.93** ± 0.04	**100.00** ± 0.0	**100.00** ± 0.0	**0.0004** ± 0.0003

- **NoConnReg (no $\mathcal{L}_{\text{conn}}$)**: This variant removes only the regularization term $\mathcal{L}_{\text{conn}}$.
- **NoStructReg (only $\mathcal{L}_X + \mathcal{L}_E + \beta_g \mathcal{L}_{\text{graphon}}$ and $\mathcal{L}_{\text{KL}}$)**: This variant removes both structural regularization terms $\mathcal{L}_{\ell_1}$ and $\mathcal{L}_{\text{conn}}$, retaining only $\mathcal{L}_{\text{KL}}$.

Table 10 shows a consistent degradation as the regularization components of Stage 1 are progressively removed. The full objective achieves the strongest overall performance on both QM9 and ZINC250K, whereas removing only $\mathcal{L}_{\ell_1}$ causes the smallest degradation, indicating that coefficient regularization provides a useful but comparatively modest refinement to the learned graphon prior. Removing $\mathcal{L}_{\text{conn}}$ produces a larger drop, particularly in validity and NSPDK, consistent with its role in encouraging a sufficiently connected source topology before Stage 2 transport. Removing both structural regularizers (NoStructReg) further degrades all four metrics, showing that $\mathcal{L}_{\ell_1}$ and $\mathcal{L}_{\text{conn}}$ provide complementary structural regularization rather than redundant constraints. The largest single-term degradation occurs when $\mathcal{L}_{\text{KL}}$ is removed: NoKLReg substantially reduces validity, uniqueness, and novelty and increases NSPDK on both datasets. This is consistent with the role of the KL term in aligning the training-time latent posteriors with the priors used at generation; without this alignment, samples drawn from $z_g \sim \mathcal{N}(0, I)$ and $p_\omega(z_i \mid z_g, s_i)$ need not lie in latent regions well represented during training, degrading the resulting source graphs and subsequent transport. Finally, removing all regularization (NoReg) yields the weakest performance throughout. Together, these results support the complementary roles of latent-prior alignment, graphon coefficient regularization, and source-connectivity regularization in Stage 1.

I.5.5 Ablation Analysis of Stage-2 Regularization Terms

The Stage-2 objective $\mathcal{L}_{S2}$ (Eq. 20) combines the velocity-matching term $\mathcal{L}_{\text{vel}}$, which is never removed, with three regularizers: $\beta_{\text{end}}\mathcal{L}_{\text{end}}$ and $\beta_{\text{val}}\mathcal{L}_{\text{val}}$, adopted from Wijesinghe et al. (2026) (Appendix G.5), and $\beta_{\text{spec}}\mathcal{L}_{\text{spec}}$ (Section 3.3), the Stage-2 spectral connectivity regularizer introduced in this work. Unlike Flowette's $\mathcal{L}_{\text{atom}}$, which stabilizes the predicted atom-type marginal, $\mathcal{L}_{\text{spec}}$ directly regularizes the algebraic connectivity of the predicted endpoint adjacency.

To evaluate the role of each Stage-2 regularizer, we conduct an ablation study by selectively removing its key elements as shown in Table 11. As in Section I.5.4, we report this ablation using the Bernstein basis only.

- **NoReg (only $\mathcal{L}_{\text{vel}}$)**: This variant removes all regularization terms and only keeps $\mathcal{L}_{\text{vel}}$.
- **NoEndReg (no $\mathcal{L}_{\text{end}}$)**: This variant removes only the regularization term $\mathcal{L}_{\text{end}}$.
- **NoValenceReg (no $\mathcal{L}_{\text{val}}$)**: This variant removes only the regularization term $\mathcal{L}_{\text{val}}$.
- **NoSpecReg (no $\mathcal{L}_{\text{spec}}$)**: This variant removes only the regularization term $\mathcal{L}_{\text{spec}}$.
- **NoAuxReg (only $\mathcal{L}_{\text{vel}}$ and $\mathcal{L}_{\text{end}}$)**: This variant removes both $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{spec}}$.

Table 11 shows a consistent degradation as the regularization terms are removed from the Stage-2 objective. The full model achieves the strongest performance on both QM9 and ZINC250K, while retaining only $\mathcal{L}_{\text{vel}}$ (NoReg) yields the weakest results, confirming that velocity matching alone is insufficient for accurate molecular graph generation. Among the individual ablations, removing $\mathcal{L}_{\text{end}}$ causes the largest overall degradation, consistent with its role as a $(1-t)^2$-reweighting of the velocity residual that concentrates supervision on the early-trajectory errors most amplified by finite-step integration (Theorem 10), rather than a constraint applied only at the terminal state. Removing $\mathcal{L}_{\text{val}}$

Table 11: Ablation study of Stage-2 regularization terms on QM9 and ZINC250K. Recall from Eq. 20, the components of the Stage-2 objective: the velocity-matching term $\mathcal{L}_{\text{vel}}$, end-to-end transport consistency $\mathcal{L}_{\text{end}}$, soft valence constraint $\mathcal{L}_{\text{val}}$, and spectral connectivity constraint $\mathcal{L}_{\text{spec}}$.

	QM9				ZINC250K			
Variant (Bernstein)	Valid↑	Unique↑	Novel↑	NSPDK↓	Valid↑	Unique↑	Novel↑	NSPDK↓
NoReg	71.51 ± 0.03	69.50 ± 0.04	69.88 ± 0.03	0.0305 ± 0.0004	76.00 ± 0.01	84.36 ± 0.04	84.50 ± 0.01	0.0340 ± 0.0003
NoEndReg	74.10 ± 0.03	81.45 ± 0.02	81.25 ± 0.05	0.0220 ± 0.0003	81.54 ± 0.04	87.07 ± 0.03	87.43 ± 0.04	0.0250 ± 0.0002
NoValenceReg	81.35 ± 0.02	92.05 ± 0.05	91.86 ± 0.05	0.0100 ± 0.0005	90.50 ± 0.06	93.06 ± 0.07	93.28 ± 0.03	0.0120 ± 0.0003
NoSpecReg	91.56 ± 0.04	96.58 ± 0.02	96.25 ± 0.05	0.0030 ± 0.0004	92.03 ± 0.02	97.95 ± 0.03	97.70 ± 0.03	0.0040 ± 0.0002
NoAuxReg	77.40 ± 0.06	87.05 ± 0.07	86.85 ± 0.05	0.0160 ± 0.0003	85.18 ± 0.01	91.05 ± 0.06	91.15 ± 0.03	0.0180 ± 0.0006
Full model	**99.93** ± 0.03	**99.65** ± 0.02	**99.25** ± 0.05	**0.0002** ± 0.0005	**99.93** ± 0.04	**100.00** ± 0.0	**100.00** ± 0.0	**0.0004** ± 0.0003

also substantially reduces validity, highlighting the importance of explicitly discouraging chemically infeasible bonding. In contrast, removing $\mathcal{L}_{\text{spec}}$ retains comparatively high uniqueness and novelty but noticeably reduces validity and increases NSPDK, indicating that spectral connectivity regularization primarily improves structural fidelity rather than sample diversity. Moreover, NoAuxReg, which retains $\mathcal{L}_{\text{end}}$ but removes both $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{spec}}$, outperforms NoEndReg, further demonstrating the importance of endpoint consistency during transport. Overall, the results support complementary roles for the three Stage-2 regularizers: $\mathcal{L}_{\text{end}}$ promotes accurate endpoint prediction, $\mathcal{L}_{\text{val}}$ encourages chemical feasibility, and $\mathcal{L}_{\text{spec}}$ preserves structural connectivity, with their combination yielding the strongest generation performance.

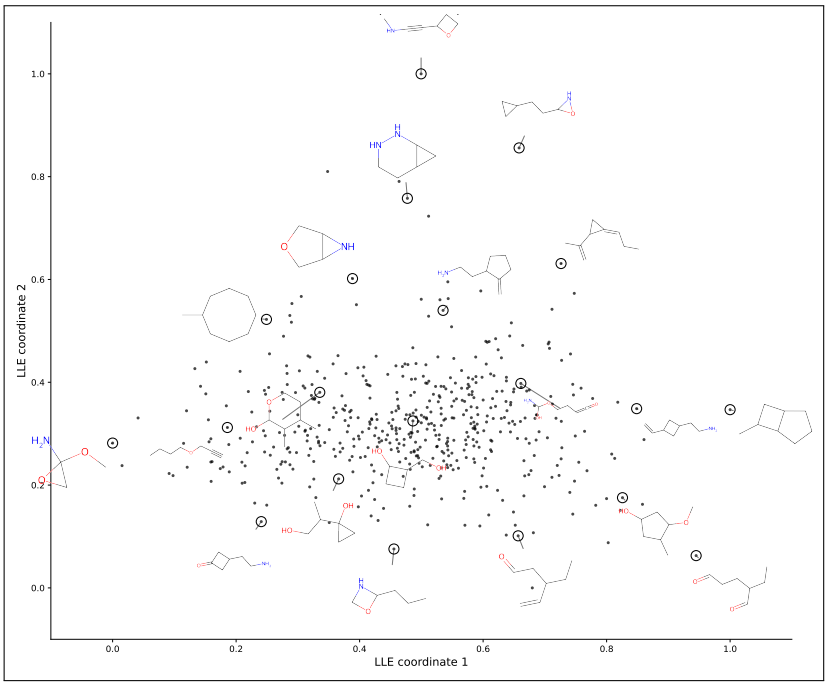
J Limitations

While *GraphonFlow* demonstrates strong performance across molecular and synthetic graph benchmarks, several aspects offer avenues for further extension: (i) canonical graphon supervision currently relies on deterministic graph labelling to establish a consistent correspondence between graph structure and graphon coordinates; although this removes the need for graph matching or learned alignment, developing canonicalisation-free graphon objectives could further simplify the framework and facilitate scaling to larger graphs; (ii) connectivity and molecular feasibility are promoted through differentiable spectral and valence regularisation, together with terminal projection, and incorporating domain-specific structural constraints directly into the continuous dynamics provides a natural direction for extending the framework to settings with additional hard combinatorial requirements; and (iii) our evaluation focuses on four molecular benchmarks and established synthetic graph families, so evaluation on larger real-world networks and broader attributed graph domains would further establish the generality of latent-conditioned graphon sources.

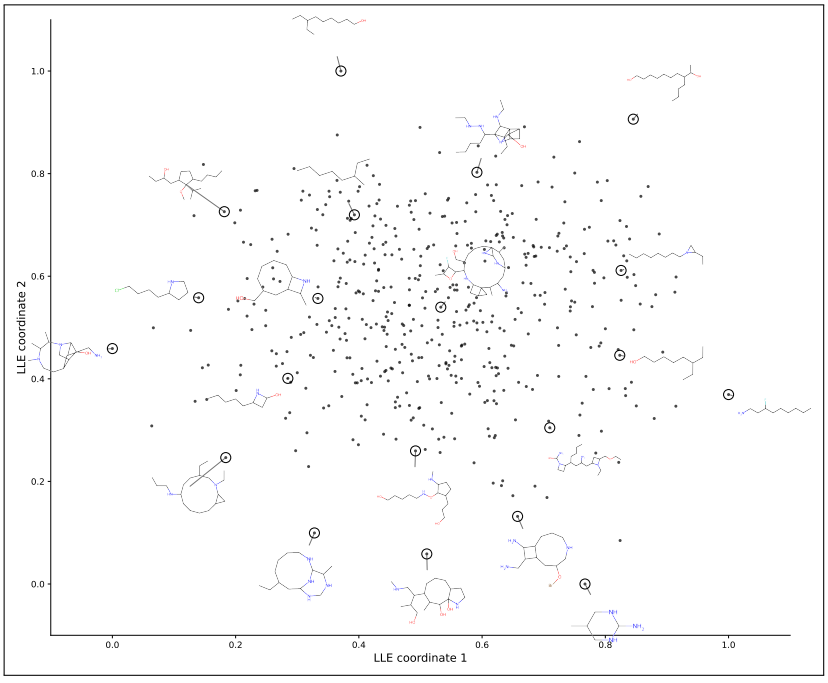
K Visualisations of Generated Graph Samples

We provide qualitative examples of molecular graphs generated by *GraphonFlow* using both graphon parameterisations. Fig. 7– 10 and Fig. 11– 14 show representative samples generated with the Fourier and Bernstein graphons, respectively, illustrating the molecular structures produced by the learned latent-conditioned graphon sources and subsequent flow-based refinement.

We further visualise synthetic graphs generated by *GraphonFlow* across the Tree, SBM, and Ego-small datasets. Fig. 15–17 and Fig. 18–20 present representative samples obtained using the Fourier and Bernstein graphon parameterisations, respectively. These examples qualitatively illustrate the diverse structural patterns produced by the learned graphon sources and subsequent flow-based refinement across the three synthetic graph families.



(a) QM9



(b) Guacamol

Figure 6: 2D LLE visualisation of the molecular representations learned by *GraphonFlow*. Each point denotes a molecule, with representative molecular structures shown next to selected points across the embedding. The QM9 representation exhibits a relatively compact organisation across small molecular structures, whereas GuacaMol spans a broader and more heterogeneous region containing diverse chain and cyclic structures. LLE is used only for two-dimensional visualisation; quantitative distributional fidelity is evaluated separately using the molecular generation metrics.

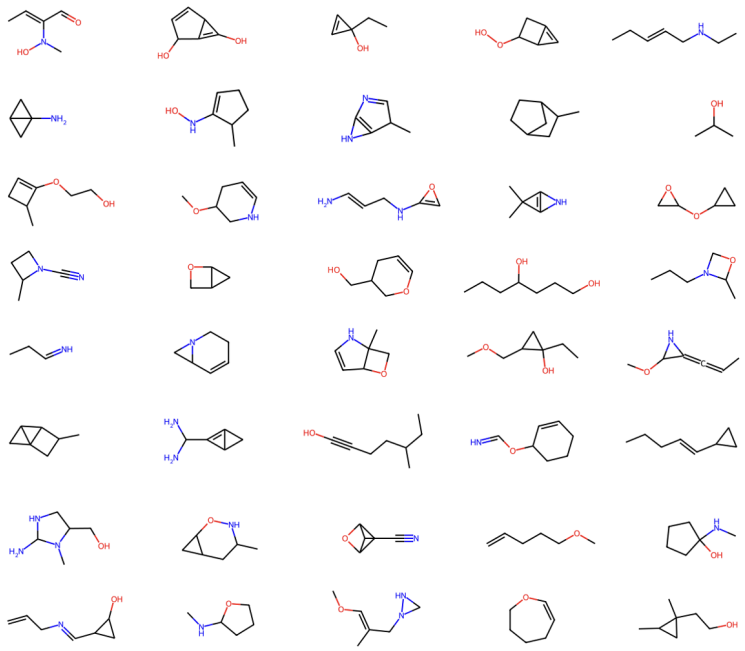


Figure 7: Molecular graph samples generated by *GraphonFlow* with the Fourier graphon parameterisation on QM9.

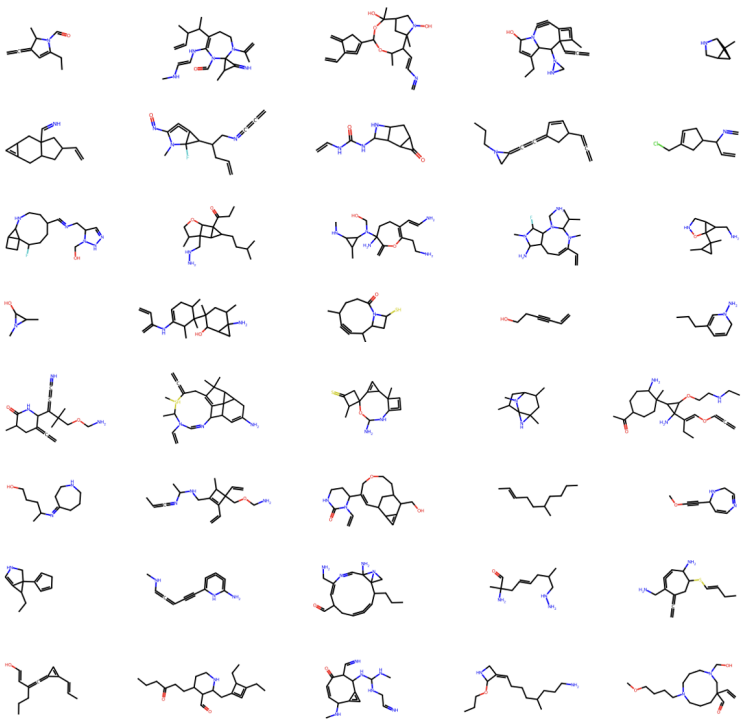


Figure 8: Molecular graph samples generated by *GraphonFlow* with the Fourier graphon parameterisation on ZINC.

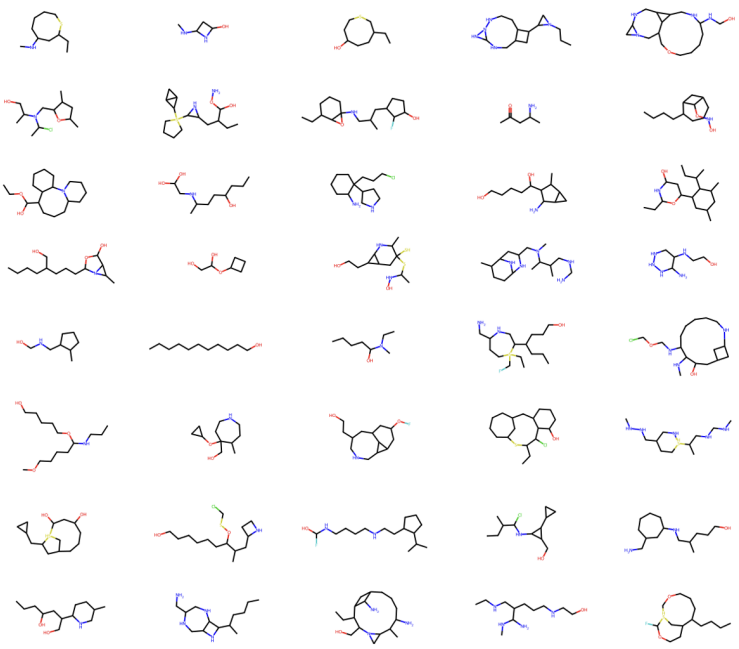


Figure 9: Molecular graph samples generated by *GraphonFlow* with the Fourier graphon parameterisation on MOSES.

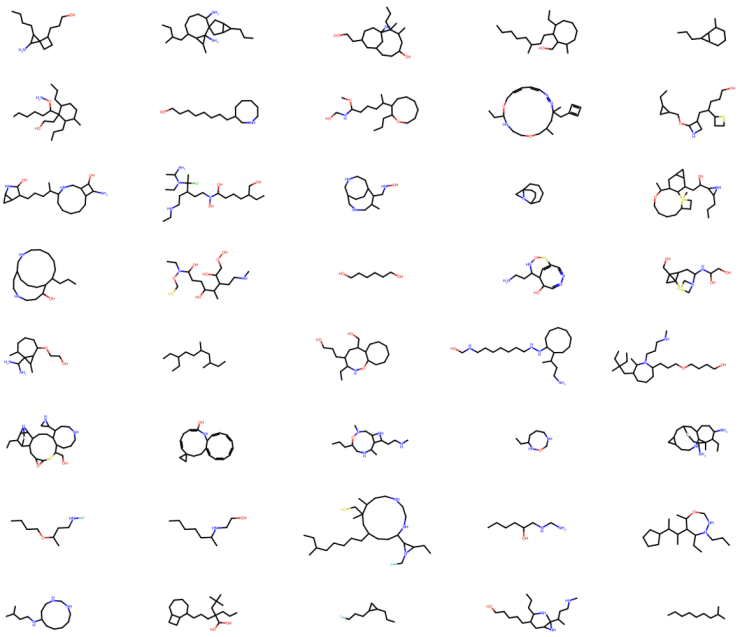


Figure 10: Molecular graph samples generated by *GraphonFlow* with the Fourier graphon parameterisation on Guacamol.

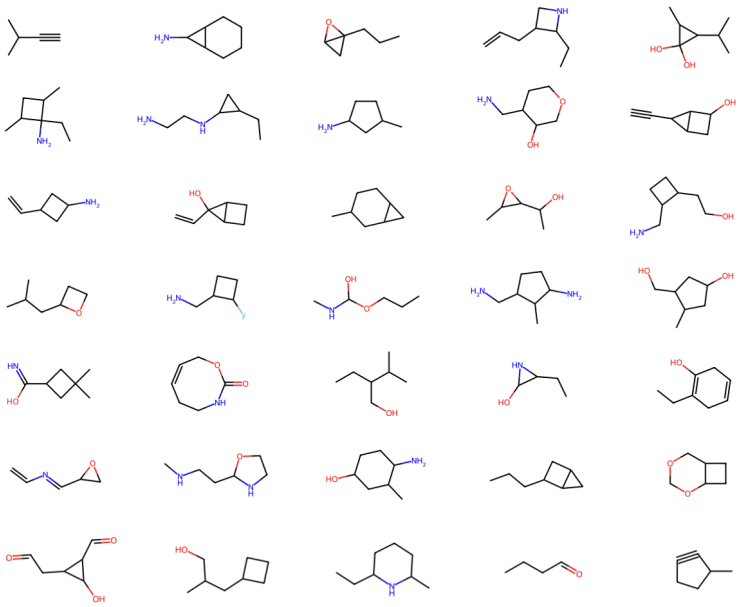


Figure 11: Molecular graph samples generated by *GraphonFlow* with the Bernstein graphon parameterisation on QM9.

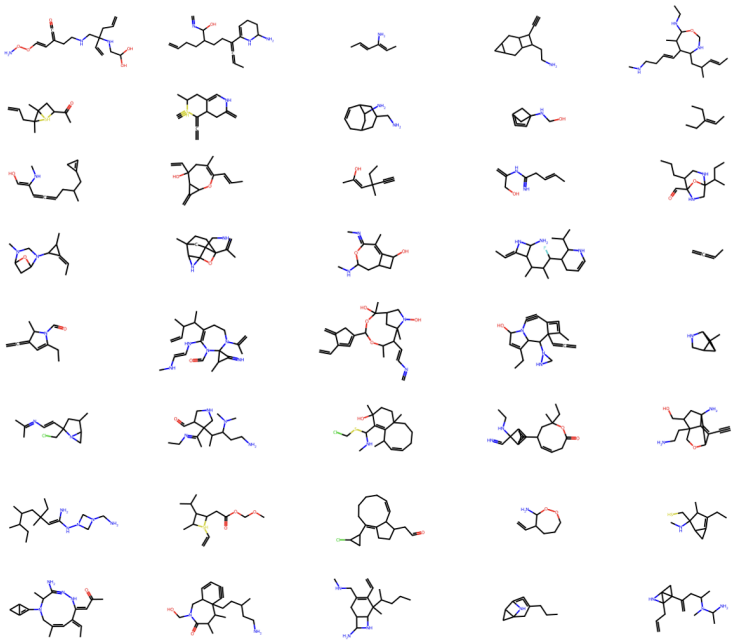


Figure 12: Molecular graph samples generated by *GraphonFlow* with the Bernstein graphon parameterisation on ZINC.

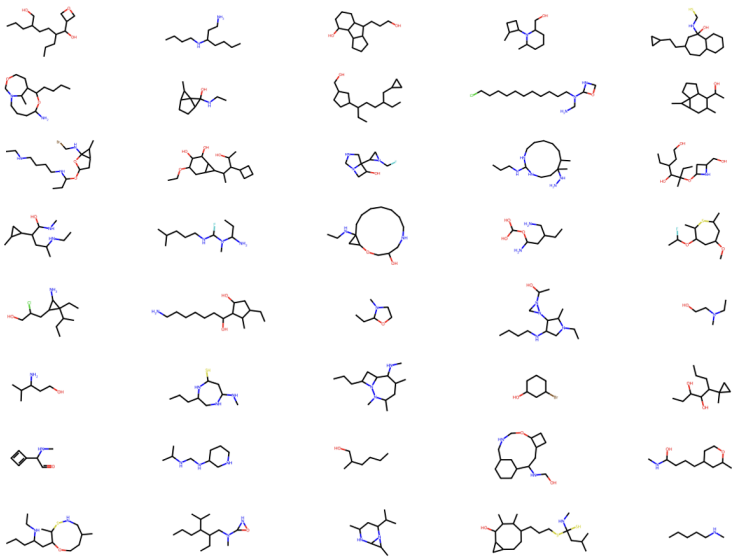


Figure 13: Molecular graph samples generated by *GraphonFlow* with the Bernstein graphon parameterisation on MOSES.

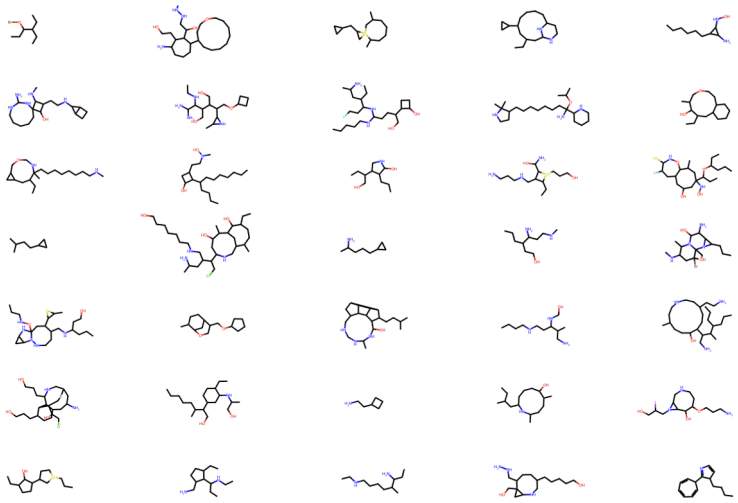


Figure 14: Molecular graph samples generated by *GraphonFlow* with the Bernstein graphon parameterisation on Guacamol.

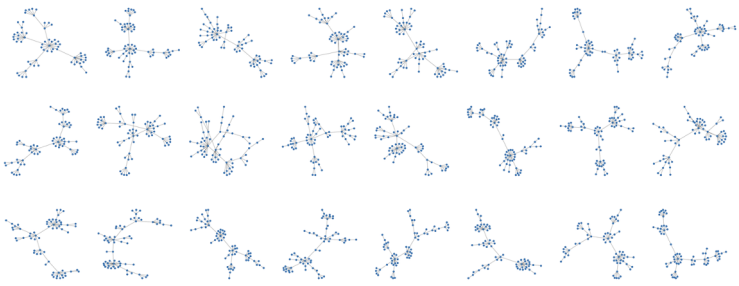


Figure 15: Synthetic graph samples generated by *GraphonFlow* with the Fourier graphon parameterisation on Tree.

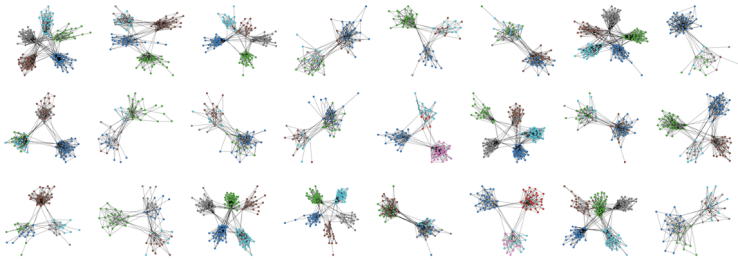


Figure 16: Synthetic graph samples generated by *GraphonFlow* with the Fourier graphon parameterisation on SBM.

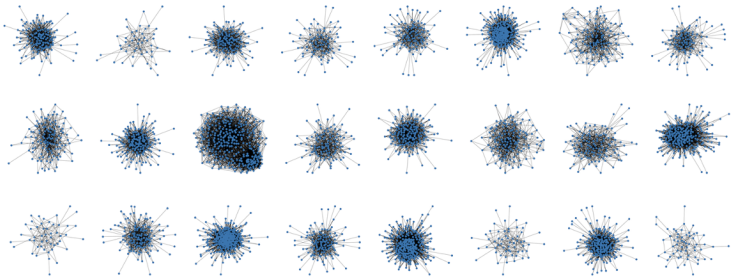


Figure 17: Synthetic graph samples generated by *GraphonFlow* with the Fourier graphon parameterisation on Ego-small.

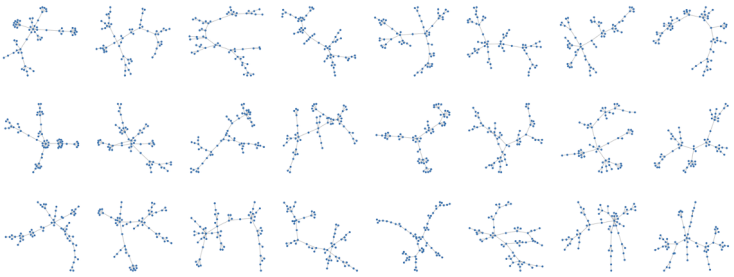


Figure 18: Synthetic graph samples generated by *GraphonFlow* with the Bernstein graphon parameterisation on Tree.

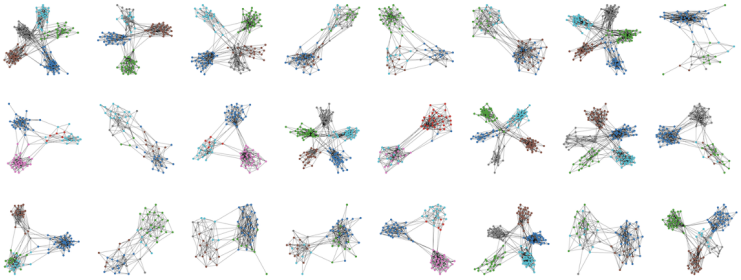


Figure 19: Synthetic graph samples generated by *GraphonFlow* with the Bernstein graphon parameterisation on SBM.

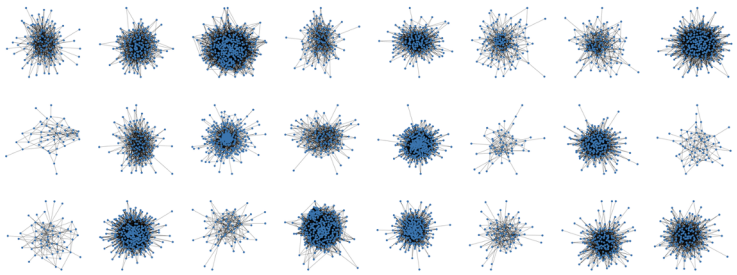


Figure 20: Synthetic graph samples generated by *GraphonFlow* with the Bernstein graphon parameterisation on Ego-small.