Permutation-Invariant Spectral Learning via Dyson Diffusion

Tassilo Schwarz[1,2] **Cai Dieball**[2,3] **Constantin Kogler**[4] **Renaud Lambiotte**[1]
Arnaud Doucet[5] **Aljaz Godec**[2,6] **George Deligiannidis**[5]
[1]Mathematical Institute, University of Oxford
[2]Mathematical bioPhysics Group, Max Planck Institute for Multidisciplinary Sciences
[3]Institute for Theoretical Physics, University of Cologne
[4]School of Mathematics, Institute for Advanced Study
[5]Department of Statistics, University of Oxford
[6]Mathematical Physics and Stochastic Dynamics, University of Freiburg
{tassilo.schwarz,lambiotte}@maths.ox.ac.uk cdieball@uni-koeln.de
kogler@ias.edu agodec@physik.uni-freiburg.de
{doucet,deligian}@stats.ox.ac.uk

Abstract

Diffusion models are central to generative modeling and have been adapted to graphs by diffusing adjacency matrix representations. The challenge of having up to $n!$ such representations for graphs with n nodes is only partially mitigated by using permutation-equivariant learning architectures. Despite their computational efficiency, existing graph diffusion models struggle to distinguish certain graph families and their spectra, unless graph data are augmented with ad hoc features. This shortcoming stems from enforcing the inductive bias within the learning architecture. In this work, we leverage random matrix theory to analytically extract the spectral properties of the diffusion process, allowing us to push most of the inductive bias from the architecture into the dynamics. Building on this, we introduce the Dyson Diffusion Model, which employs Dyson's Brownian motion to capture the spectral dynamics of an Ornstein-Uhlenbeck process on the adjacency matrix. Furthermore, conditioned on the spectral dynamics, we formulate a Lie group diffusion, appropriately modeling the remaining degrees of freedom. Strikingly, the resulting learning problem becomes permutation invariant at the Lie algebra level. We demonstrate that the Dyson Diffusion Model learns graph spectra accurately and outperforms existing graph diffusion models.

1 Introduction

Diffusion models are a key class of generative models based on noising data with stochastic differential equations (SDEs) and learning their time reversal [1, 2, 3]. They provide state-of-the-art results in many domains such as audio [4] and vision [5]. Generalizing diffusion models from Euclidean space to graphs offers promising applications in numerous areas, such as biology [6] or combinatorial optimization [7]. However, while diffusing adjacency matrix representations is straightforward and popular [8, 9, 10], this approach faces a major obstacle: Each graph with n vertices has up to $n!$ representations as adjacency matrices. Therefore, if one aims to use a diffusion model on the space of matrices, one must learn $n!$ representations *per graph*. This is not feasible. Previous works tackled this problem by shifting the inductive bias of permutation invariance to the learning algorithm: If the neural network was permutation equivariant, training on one of the (up to $n!$ many) matrix representations would suffice. For example, Niu et al. [8] and Jo et al. [9] build on message-passing

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

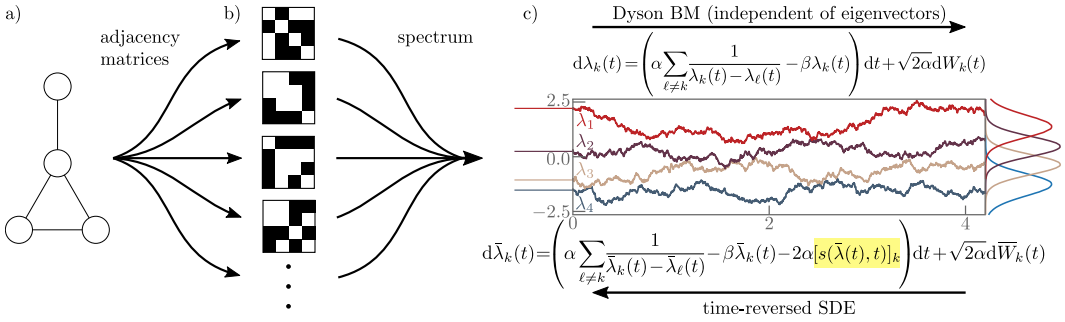


Figure 1: Dyson Diffusion Model and its application to graph spectra: A graph on n vertices (a) has up to $n!$ representations as adjacency matrices (b). For an OU-driven diffusion on *any* adjacency matrix, the permutation-invariant spectrum (c) evolves according to the same SDE (Dyson-BM). A sample path of the n non-intersecting eigenvalues is shown. The marginals of the invariant density for the λ_k are depicted on the far right. The DyDM diffusion model learns the score $s(\lambda, t)$ (highlighted in yellow) to generate spectra via the time-reversed SDE (3).

graph neural networks (GNNs) while ConGress [10] is centered around graph transformers. However, GNNs have a proven "blind spot", as detailed below.

Theoretical Limitations. The "blind spot" arises from the limited ability of these models to solve Graph Isomorphism (GI): determining if two graphs are structurally identical regardless of node labeling. While permutation equivariance ensures that the model produces consistent outputs for different representations of the same graph, it does not guarantee that the model can differentiate between two structurally different (non-isomorphic) graphs. The failure is a result of how these architectures aggregate information: permutation-symmetric operations – message passing in GNNs – can collapse distinct graphs with similar neighborhoods to the same representation, treating them as identical. More formally, GI is a challenging problem in algorithm theory, and it remains unknown whether $\mathrm{GI} \in \mathsf{P}$ [11]. Since a polynomial time (learning) algorithm perfectly distinguishing all graphs would prove $\mathrm{GI} \in \mathsf{P}$, contemporary (polynomial-time) graph learning algorithms must compromise on expressivity, which arises when graphs (and noisy perturbations thereof) are sufficiently indistinguishable.

Extracting Permutation-Invariant Information from Graph Diffusion. When diffusing an entire adjacency matrix, state-of-the-art work pushes the entire inductive bias into the learning algorithm [8, 9] with possible data augmentation [12, 10, 13]. However, as we show below, this is neither necessary nor desirable (see the previous discussion on blind spots). Using techniques from random matrix theory, we show that an Ornstein–Uhlenbeck (OU) diffusion on the graph can be *dissected* into diffusion of the (permutation invariant) spectrum and diffusion of the (permutation-dependent) eigenvectors. Our method therefore allows us to learn the spectrum while preserving all remaining information. Moreover, since the spectrum is inherently permutation-invariant, we can parameterize the score using a much broader range of learning architectures, expanding the scope to architectures able to distinguish between graphs that are equivalent in the Weisfeiler-Leman (WL) sense [14, 15].[1]

Information in the Spectrum. The spectrum of a graph encodes key structural features including connectivity, expansion, and subgraph patterns. Moreover, non-isomorphic WL-equivalent graphs typically have distinct spectra (see e.g. [16] for regular graphs). An idea, therefore, is to augment the graph data based on spectral information [10, 13]. Our work is based on an entirely different method, exploiting analytical insight from random matrix theory to dissect the spectral from the remaining information, allowing to push the inductive bias from the architecture to the dynamics.

Dyson's Brownian Motion. For an OU process on the space of symmetric matrices, the eigenvalues follow a well-characterized stochastic differential equation, the so-called Dyson Brownian motion (DBM), which is inherently permutation invariant. Thus – in contrast to [8, 9, 10] – the score of DBM

[1] We give a theoretical discussion in form of the WL-equivalence class in Section 2 and demonstrate the challenge of those architectures empirically in Figure 2.

can be parameterized with any (not necessarily permutation invariant) neural network. Moreover, contrary to [17], the remaining information is not lost (see Theorem 3).

Contributions. Our main contributions are:

- We introduce the Dyson Diffusion Model (DyDM) in Section 3, extracting the spectral dynamics from an OU-driven diffusion. DyDM learns graph spectra without GNNs or graph transformers; the accompanying conditional eigenvector SDE provides a principled route to modeling the remaining degrees of freedom, which we validate in controlled experiments.
- We demonstrate in Section 4 that building on this SDE, DyDM is more effective than existing GNN-based and graph-transformer-based methods for learning graph spectra.
- We also derive the eigenvector SDE and its time reversal. Expressing the dynamics on the orthogonal Lie group $O(n)$ enables efficient simulation. We prove a symmetry, exponentially reducing the learning space, and translate the equivariant learning problem for eigenvectors into an invariant learning problem on the Lie algebra, opening the field to new architectures.
- Combining spectral and conditional eigenvector learning, we achieve end-to-end graph generation, without the need for GNNs or feature augmentation.

Beyond graphs, our framework generally applies to symmetric matrices where spectral information is often key. For instance, in statistics, it can encode the importance of principal components [18], while in dynamical systems,[2] it reflects the stability and timescales of linear operators.

Notation. We work on the set of symmetric real matrices $\mathrm{Sym}(\mathbb{R}^{n\times n}) := \{A \in \mathbb{R}^{n\times n} : A^T = A\}$. The positive integers up to n are denoted by $[n] := \{1, \dots, n\}$. We consider undirected graphs $G = (V, E, w)$ where V is a finite set with edges $E \subseteq \{S \subseteq V : 1 \leq |S| \leq 2\}$ allowing for self-loops and weights $w : E \to \mathbb{R}$. The family of such graphs of size n is $\mathcal{G}^n := \{G = (V, E, w) : |V| = n\}$. For a graph $G \in \mathcal{G}^n$ with $\mathbb{R}^V = \{f : V \to \mathbb{R}\}$ being the space of functions from V to $\mathbb{R}$, we let $A \equiv A(G) : \mathbb{R}^V \to \mathbb{R}^V$ be the adjacency *operator*, defined for $f \in \mathbb{R}^V$ and $v \in V$ as the weighted sum of f applied to the neighbors of v as $Af(v) \coloneqq \sum_{\{u,v\}\in E} w(\{u, v\})\, f(u)$. As the graphs are undirected, the adjacency operator is self-adjoint with respect to the standard inner product on $\mathbb{R}^V$. The operator A therefore has n real eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. We denote the ordered spectrum of the adjacency operator by $\lambda(G) \coloneqq \lambda(A(G)) \coloneqq \{(\lambda_1, \lambda_2, \dots, \lambda_n) : \lambda_1 \geq \ldots \geq \lambda_n\}$. Unless defined explicitly otherwise, we use $n \in \mathbb{N}$ for the (vertex) size of the graph, and $N \in \mathbb{N}$ for the number of samples. We denote by $s \sim \mathcal{N}(0, I_d)$ that the d-dimensional random vector s has multivariate normal distribution with 0 mean and unit covariance I_d. The orthogonal group in dimension n is denoted $O(n)$ and its Lie algebra $\mathfrak{o}(n)$ with basis elements denoted by $E_{(\ell,k)}$ (which is 1 at the (ℓ, k)-entry, -1 at (k, ℓ), and 0 otherwise).

2 Limitations of Existing Diffusion Models

We consider the matrix-valued OU-SDE starting from some data $M(0) \in \mathrm{Sym}(\mathbb{R}^{n\times n})$ given by

$$\mathrm{d}M_{ji}(t) = \mathrm{d}M_{ij}(t) = -\beta M_{ij}(t)\mathrm{d}t + D_{ij}\mathrm{d}B_{ij}(t), \quad 1 \leq i \leq j \leq n \tag{1}$$

with $D_{ij} \coloneqq \sqrt{(1+\delta_{ij})\alpha}$ for any constants $\alpha, \beta \in \mathbb{R}^+$, where $B_{ij}(t) = B_{ji}(t)$ are independent Brownian motions and $\delta_{ij} = 1$ if and only if $i = j$, and 0 otherwise. We consider (1) on the space of symmetric matrices to represent undirected graphs $G \in \mathcal{G}^n$. Equation (1) is an entry-wise OU process that preserves the symmetry of the matrix. Standard diffusion models run (1) until $T > 0$ from data samples. To fix notation, denote by p_t the distribution of $M(t)$ induced by (1) for $t \geq 0$.

The time-reversal of (1) on time interval $[0, T]$ is a diffusion initialized from $\bar{M}(T) \sim p_T$ satisfying

$$\mathrm{d}\bar{M}_{ij}(t) = -\left\{\beta \bar{M}_{ij}(t) + D_{ij}^2 \left[s(\bar{M}(t), t)\right]_{ij}\right\}\mathrm{d}t + D_{ij}\mathrm{d}\bar{B}_{ij}(t), \quad 1 \leq i \leq j \leq n \tag{2}$$

for independent Brownian motions $\bar{B}_{ij}$ [20, 2]. In (2), $s(M, t)$ represents the score matrix at time $t \in (0, T]$, i.e., $s(M, t) \coloneqq \boldsymbol{\nabla}_M \ln p_t(M)$. This score is intractable but, as the OU process has tractable Gaussian transition densities, we can obtain an estimate $s_\theta(M, t)$ of it by minimizing the denoising score matching loss [2] $L(\theta) = \mathbb{E}\left[\left\|s_\theta(M(t), t) - \boldsymbol{\nabla}_{M(t)} \ln p_{t|0}(M(t)|M(0))\right\|_2^2\right]$,

[2] For instance, the generator of Markov dynamics under detailed balance is self-adjoint, see [19].

where the expectation is over $t \sim \mathcal{U}[0,T], M(0) \sim p_0, M(t) \sim p_{t|0}(\cdot|M(0))$. Samples of p_0 can then be generated by simulating an approximation of (2) obtained by sampling $\bar{M}(T)$ from the invariant distribution of (1), $M_{ij}^{\text{inv}} \sim \mathcal{N}(0, \alpha(1+\delta_{ij})/2\beta)$, and using $s_\theta(M,t)$ in place of $s(M,t)$.

Challenges posed by graphs. When working with graphs, to obtain an adjacency matrix for a given weighted graph $G = (V, E, w) \in \mathcal{G}^n$, we need to fix an ordering $(v_1, \dots, v_n)$ of all vertices. In fact, given the ordering $(v_1, \dots, v_n)$, the associated matrix $M = (m_{ij})_{1 \le i,j \le n} \in \text{Sym}(\mathbb{R}^{n \times n})$ has entries $m_{ij} = w(\{v_i, v_j\})$ if $\{v_i, v_j\} \in E$ and 0 otherwise. The challenge is that the adjacency operator $A(G)$ for a given graph $G \in \mathcal{G}^n$ admits up to $n!$ distinct adjacency matrices. For instance, in Figure 1 we see four different matrix representations of the same graph.

Why should we enforce this inductive bias? One could think that the reason for using the inductive bias stems from wanting that graph generative models assign uniform probability to each of the (up to) $n!$ many representations. However, one may easily apply a random independent permutation to the output of the generative model. Instead, the challenge stems from the following problem: In diffusion models, we learn a function (the score) on a set of graphs, say[3] Ω, rather than the distribution directly. Learning on the adjacency representations would correspond to learning on $\Omega \times S_n$. We demonstrate on a toy example in Theorem 7 that *not* leveraging the inductive bias, i.e. learning on a space of size $\Omega \times S_n$, leads to an explosion of the average mean squared error; i.e., learning a function on a fixed number of k objects (say unweighted graphs on n nodes) from N samples yields a mean squared error of order $\Theta(n!/N)$. In contrast, using the inductive bias provides an average mean squared error of $\Theta(1/N)$. Given that $n! > 3 \cdot 10^6$ even for $n = 10$ nodes, utilizing the inductive bias is essential.

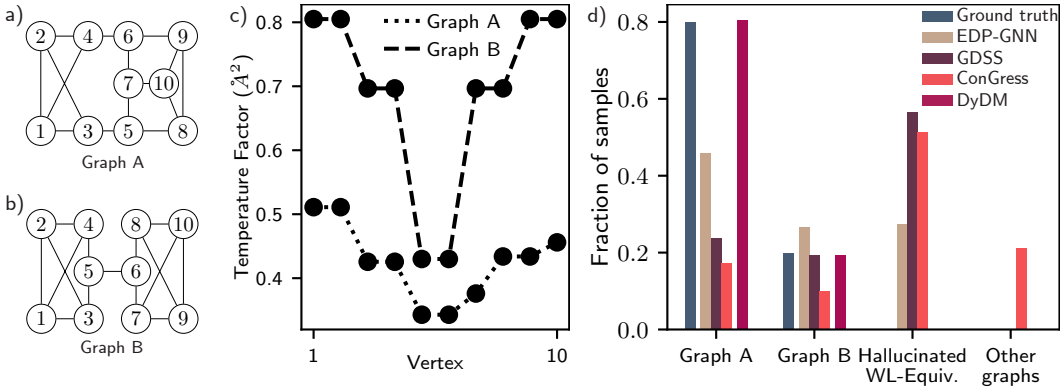


Figure 2: Example for struggle of GNN-based and graph-transformer-based models with two WL-equivalent graphs: Graph A and Graph B are WL-equivalent, but non-isomorphic. Also physically, they have very different properties, such as different temperature factors (c) and a different cut size. Upon training on an 80% Graph A and 20% Graph B dataset, state-of-the-art GNN-based (EDP-GNN, GDSS) and graph-transformer-based (ConGress) models learn the WL-equivalence class quickly but fail to generate the underlying distribution among the two graphs, with some even predominantly hallucinating WL-equivalent but non-isomorphic graphs (d).

Cost of pushing inductive bias entirely into architecture. One solution would be to impose the inductive bias in the learning architecture. This is what state-of-the-art graph diffusion models do [8, 9, 10]. However, using these architectures comes at a cost. As argued in the introduction, since $\text{GI} \in \text{P}$ remains unknown, we expect some limitations. In the case of GNNs this compromise has been precisely characterized: GNNs cannot distinguish between the large families of so-called Weisfeiler-Leman (WL) equivalent graphs [14, 15]. For example, all k-regular graphs on n vertices for any fixed $k \in \mathbb{N}$ are WL-equivalent (see Lemma 1 below and its proof in App. N). This limitation may lead to hallucination and is also evident in the learned spectra.

Lemma 1 (WL-equivalence of k-regular graphs)**.** *For fixed $n, k \in \mathbb{N}$, all k-regular graphs $G \in \mathcal{G}^n$ are WL equivalent. Every graph $G \in \mathcal{G}^n$ that is WL equivalent to a k-regular graph is k-regular.*

This is a vast class, since e.g. on $n = 20$ vertices, there are 510,489 such non-isomorphic, connected 3-regular and thereby WL-equivalent graphs [21]. In particular, we demonstrate on a simple example

[3] Importantly, Ω refers to *the set of graphs* and *not* the set of (permutation-sensitive) adjacency matrices.

in Figure 2 that both GNN- and graph-transformer-based methods fail to learn a distribution on two particular WL-equivalent graphs: during sampling, they either fail to learn the distribution on both WL-equivalent graphs or hallucinate other, WL-equivalent but non-isomorphic, graphs (Fig. 2d). This can also be observed during learning: The EDP-GNN model learns the WL-equivalence class quickly (after 500 epochs) but then keeps hallucinating non-isomorphic but WL-equivalent graphs, and in particular struggles to learn the right distribution between graphs A and B for the remaining 4500 epochs (see App. R for details). Importantly, those graphs are very different. For instance, if the graphs represented Gaussian Network Models for macromolecules [22], physical observables such as the temperature factors in X-ray scattering [23] would be clearly distinct, see Fig. 2c. Therefore, a diffusion model based on GNNs will suffer from this expressivity blind spot. More generally, any diffusion model relying on such a graph-specific learning algorithm will have limited expressivity.

3 Dyson Diffusion Model

3.1 Dyson Brownian Motion

Dyson [24] showed that the spectrum of (1) behaves as n positively charged particles in a one-dimensional Coulomb gas. These particles exhibit Brownian motion, but with a pairwise repulsion force proportional to their inverse distance so that their paths do not cross (see Fig. 1c). More precisely, Dyson proved that the spectrum of the entry-wise Ornstein–Uhlenbeck process from (1) follows the SDE given in Theorem 2. Without loss of generality, we restrict the domain to the *Weyl Chamber* $C_n := \{\lambda \in \mathbb{R}^n : \lambda_1 > \ldots > \lambda_n\}$.

Theorem 2 (Eigenvalue SDE [24])**.** *Denote by $\lambda(t) = (\lambda_1(t), \ldots, \lambda_n(t))$ the ordered spectrum of the matrix-valued Ornstein-Uhlenbeck process $M(t)$ of* (1)*. Then assuming that the initial matrix $M(0)$ has simple spectrum, $\lambda(t)$ satisfies for all $1 \leq k \leq n$ the stochastic differential equation*

$$\mathrm{d}\lambda_k(t) = \left(\alpha \sum_{\ell \neq k} \frac{1}{\lambda_k(t) - \lambda_\ell(t)} - \beta \lambda_k(t) \right) \mathrm{d}t + \sqrt{2\alpha} \mathrm{d}W_k(t), \qquad \text{(Dyson-BM)}$$

for $W_1, \ldots, W_n$ independent standard Brownian motions. Moreover, the unique stationary distribution of (Dyson-BM) *has density $p_{\mathrm{inv}}(\lambda) = \exp(-U(\lambda))/Z$ for $U(\lambda) = \frac{\beta}{2\alpha} \sum_k \lambda_k^2 - \sum_{k<\ell} \ln|\lambda_k - \lambda_\ell|$, $\lambda \in C_n$, and Z a normalizing constant so that p_{inv} corresponds to a probability measure.*

For completeness, a full proof of Theorem 2 is given in App. B, where we generalize a well-known proof to arbitrary coefficients α, β. We note that the assumption of Theorem 2 that M_0 has simple spectrum is minor. Indeed, generic random graphs or matrices have simple spectra, and in the case of eigenvalues with higher multiplicity, we can perturb the spectrum to be simple. From (Dyson-BM) we see that the eigenvalues perform a Brownian motion in a confining potential with a repulsion force: once a pair of eigenvalues λ_k, λ_l comes too close, they become repelled with a force $\alpha/(\lambda_k - \lambda_l)$ inversely proportional to their separation. A remarkable property of Theorem 2 is that the evolution of the spectrum is decoupled from all other information about the matrix: the spectral SDE (Theorem 2) is independent of the eigenvectors. This is the key analytical insight that motivates our approach. Furthermore, conditioned on the eigenvalues, the remaining information captured in form of the eigenvectors can be deduced as we show in Theorem 3. This generalizes a statement of Allez et al. [25], and we give a proof in App. C.

Theorem 3 (Eigenvector SDE)**.** *Denote by $(v_1(t), \ldots, v_n(t))$ the orthonormal eigenvectors associated to the eigenvalues of Theorem 2. Assuming that the initial matrix $M(0)$ has simple spectrum, $v_k(t)$ satisfies for $k \in [n]$ the stochastic differential equation*

$$\mathrm{d}v_k(t) = -\frac{\alpha}{2} \sum_{\ell \neq k} \frac{v_k(t)}{(\lambda_k(t) - \lambda_\ell(t))^2} \mathrm{d}t + \sqrt{\alpha} \sum_{\ell \neq k} \frac{v_\ell(t)}{\lambda_k(t) - \lambda_\ell(t)} \mathrm{d}w_{\ell k}(t) \qquad \text{(Eigenvector-SDE)}$$

for $\{w_{ij:i\neq j}\}$ standard Brownian motions independent of the eigenvalue trajectories, with $w_{ji} = w_{ij}$.

Given that $(v_1(t), \ldots, v_n(t))$ are orthonormal, we observe for (Eigenvector-SDE) the following properties. For the norm of any component $k \in [n]$, and the scalar product for any $l \neq k$, we have

$$[v_k(t) + \mathrm{d}v_k(t)]^2 = 1 + 2v_k^T(t)\mathrm{d}v_k(t) + [\mathrm{d}v_k(t)]^2 = 1 + \mathcal{O}\left(\mathrm{d}t^{3/2}\right)$$

$$[v_k(t) + \mathrm{d}v_k(t)]^T [v_l(t) + \mathrm{d}v_l(t)] = 0 + v_k(t)^T \mathrm{d}v_l(t) + v_l^T(t)\mathrm{d}v_k(t) + \mathrm{d}v_k^T(t)\mathrm{d}v_l(t) = \mathcal{O}\left(\mathrm{d}t^{3/2}\right),$$

and refer for details to Appendix C. Thus, (Eigenvector-SDE) inherently preserves orthonormality. Any learning algorithm can therefore *focus on the actual distribution* on the manifold, and does not have to learn the manifold $O(n)$ within $\mathbb{R}^{n\times n}$ itself.

3.2 From Dyson SDE to Spectral Diffusion Model

The analytical expression in Theorem 2 allows learning the spectrum of a matrix-diffusion following (1). Despite its advantages, constructing a diffusion model based on (Dyson-BM) poses several challenges, e.g. dealing with a singular drift, non-Gaussian conditional density, etc. (see App. L for details). As described below, with DyDM we overcome these obstacles and design an efficient diffusion model for the spectrum, which can distinguish between spectra of graphs that GNNs are blind to (Fig. 2) and which does *not* require ad hoc data augmentation.

The time-reversal of (Dyson-BM) in the sense of [20] reads

$$\mathrm{d}\bar{\lambda}_k(t) = \left[\alpha \sum_{\ell\neq k} \frac{1}{\bar{\lambda}_k(t) - \bar{\lambda}_\ell(t)} - \beta\bar{\lambda}_k(t) - 2\alpha[s(\bar{\lambda}(t),t)]_k\right]\mathrm{d}t + \sqrt{2\alpha}\mathrm{d}\overline{W}_k(t), \tag{3}$$

where we aim to learn the score $s(\lambda,t) \coloneqq \boldsymbol{\nabla}_\lambda \ln p_t(\lambda)$. As the coefficients in (Dyson-BM) are non-Lipschitz, the applicability of [20] is not immediate. Accordingly, in App. D we present a proof of existence and uniqueness of a strong solution and verify that Anderson's time reversal applies.

Making the loss tractable. Learning the score $s(\lambda,t)$ from the loss $L(\theta)$ (as defined in Section 2) is not feasible for the (Dyson-BM), since the conditional distribution $p_{t|0}$ is not available in closed form in contrast to the OU process. To overcome this, we follow a derivation in the style of [26] to obtain the loss function – up to constants in θ – for any $h \in \mathbb{R}^+$

$$\tilde{L}(\theta) = \mathbb{E}\left[\left\| s_\theta(\lambda_{t+h}, t+h) - \boldsymbol{\nabla}_{\lambda_{t+h}} \ln p_{t+h|t}(\lambda_{t+h}|\lambda_t)\right\|_2^2\right], \tag{4}$$

the expectation being over $t \sim \mathcal{U}[0,T]$, $\lambda_t \sim p_t$, $\lambda_{t+h} \sim p_{t+h|t}(\cdot|\lambda_t)$. We approximate $p_{t+h|t}$ with the Gaussian transition of the Euler–Maruyama step, which is exact for $h \to 0$ (see App. A).

Handling singularities. Numerical solutions of (Dyson-BM) with a fixed step size are not practical, since (Dyson-BM) is singular at the boundary of the Weyl Chamber. A fixed step size leads to inaccuracies at the boundaries and may overshoot the singularity, leaving the Weyl Chamber. To overcome this, we implement an adaptive step-size algorithm which conditions on an event of probability 1 (non-crossing) and hence does not change marginal densities.[4] The step-size controller is described in Algorithm 2 and in App. F.1 in detail.

Schedule. Dyson's conjecture states that for $\alpha = \frac{1}{n}$ and $\beta = \frac{1}{2}$ (Dyson-BM) converges to the invariant distribution in time $\Theta(1)$, while the majority of eigenvalues mixes *locally* already in time $\Theta(1/n)$ [27]. Hence, the fine-grained structure will be mixed in the time interval $(0, \Theta(1/n))$. We show in App. E through time change that this conjecture can be applied to any choice of α, β. It is thus sensible to choose an exponential schedule. The particular choices are specified in App. K.1.

Equilibrium shooting mechanism. In the backward dynamics, imperfect learning may erroneously lead to crossing of the Weyl Chamber. To overcome this, we introduce a repulsion force deduced from the equilibrium distribution, acting close to the boundary. For details, see App. G.

Sampling from Invariant Distribution. To sample from the invariant distribution of $\lambda(t)$ given by p_{inv} we exploit the connection between $\lambda(t)$ and $M(t)$: We first sample from the Gaussian invariant distribution of $M(t)$ and then perform an eigendecomposition.

Comparison to direct simulation. There are clear efficiency and precision reasons for why we do not sample from (1) in combination with eigendecompositions, see App. G.1. Moreover, our SDE implementation renders the training runtime on the order of a simulation-free diffusion model.

3.3 Eigenvector Time Reversal and Numerical Scheme

While DyDM focuses on learning spectra, a defining feature is that the eigenvector information remains accessible. We now present the time reversal and numerical scheme for the (Eigenvector-SDE), building on existing work on Lie group diffusion [26, 28].

[4] The adaptive step-size is only used in the forward simulation. The objective is evaluated on a fixed grid for correctness.

Algorithm 1 DyDM training

Input spectral samples $\lambda^{(1)}_{\cdot}, \ldots, \lambda^{(N)} \in \mathbb{R}^n_{\cdot}$ schedule $\mathcal{T}=\{t_j\}$

for each sample $i \in [N]$ **do**
  $t \leftarrow 0$
  $\lambda(t) \leftarrow \lambda^{(i)}$
  while $t < T$ **do** ▷ Diffuse sample
    Let $u \sim \mathcal{N}(0, I_n)$
    $\delta t \leftarrow$ ForwardStepsizeController$(\lambda(t), u)$ ▷ Conditions on non-intersection, see F.1.
    if $\delta t < \delta t_{\text{min}}$ **then**
      Skip step ▷ Rare event: Skip tiny steps
    end if
    $t_{\text{fix}} = \min\{t_j \in \mathcal{T} : t_j > t\}$
    $\delta t \leftarrow \min\{\delta_t, t_{\text{fix}} - t\}$
    $\lambda(t+\delta t) \leftarrow$ Euler-Maruyama step of (Dyson-BM) with step size δt and noise u
    $t \leftarrow t + \delta t$
  end while
end for
Update s_θ using loss $\tilde{L}(\theta)$ along paths $\lambda^{(1)}(t), \ldots, \lambda^{(N)}(t)$ on schedule $\mathcal{T}$ using Eq. (19)

Output s_θ.

Figure 3: Dyson Diffusion Model (training): The (Dyson-BM) is evolved forward in time with an adaptive step size ensuring that the paths remain in the Weyl Chamber. The step-size controller conditions on the probability 1 event of non-crossing as detailed in App. F.1 and Algorithm 2.

Denoting the matrix whose columns are the eigenvectors as $X_t = (v_1(t), \ldots, v_n(t)) \in \mathbb{R}^{n\times n}$, and with $\mathrm{d}W^{(\ell,k)}$ being i.i.d. Wiener increments, we may rewrite the SDE as follows.

Proposition 4. *The following* $\mathrm{O}(n)$ *Stratonovich SDE is equivalent to* (Eigenvector-SDE)

$$dX(t) = \sqrt{\alpha} \sum_{1\le\ell<k\le n} \left(\frac{X(t)E_{(\ell,k)}}{\lambda_k(t) - \lambda_\ell(t)} \right) \circ \mathrm{d}W^{(\ell,k)}(t). \tag{5}$$

Thus, we can derive the numerical update step following Ref. [29] for a small increment as

$$X(t + \mathrm{d}t) \approx X(t) \exp(\mathrm{d}Z(t)), \quad \mathrm{d}Z(t) = \sqrt{\alpha} \sum_{1\le\ell<k\le n} \frac{E_{(\ell,k)}}{\lambda_k(t) - \lambda_\ell(t)} \mathrm{d}W^{(l,k)}(t) \in \mathfrak{o}(n) \tag{6}$$

Since $\mathrm{d}Z(t) = -\mathrm{d}Z(t)^T \in \mathfrak{o}(n)$, the matrix exponential form (6) shows directly the preservation of orthogonality $X(t + \mathrm{d}t) \in \mathrm{O}(n)$. The advantage of this form is that the deviation from the Lie group only arises due to precision errors of matrix exponentiation, and not from the time discretization.

The Eigenvectors' time reversal[5] is $\bar{X}(t + \mathrm{d}t) = \bar{X}(t) \exp\big(\mathrm{d}\bar{Z}(t)\big)$ with Lie algebra increment

$$\mathrm{d}\bar{Z}(t) = -\alpha \nabla \ln p_t(\bar{\lambda}, \bar{X})\mathrm{d}t + \sqrt{\alpha} \sum_{1\le\ell<k\le n} \frac{E_{(\ell,k)}}{\bar{\lambda}_k(t) - \bar{\lambda}_\ell(t)} \mathrm{d}\bar{W}^{(\ell,k)}(t) \in \mathfrak{o}(n). \tag{7}$$

Phrasing the forward- and backward-dynamics, *including the score*, via Lie *algebra* increments $\mathrm{d}Z_t, \mathrm{d}\bar{Z}_t$ offers beyond numerical precision another key advantage: The number of degrees of freedom is $n(n-1)/2$ instead of the naive n^2, as is apparent from the summation over $\ell < k$. Strikingly, it enables using a permutation-invariant learning architecture. While permutation equivariance of the columns of X_t is already resolved, since they correspond to the canonically ordered eigenvalues, permutation equivariance of the rows of X_t is covered by the following Lemma 5. This represents two levels of simplification, since permutation equivariance of both, rows and columns, is simplified to fixed columns and permutation-invariance in the rows, as we prove in App. S.5.

[5] For details see App. S.3 and [26] for related results for Brownian dynamics on a manifold.

Table 1: Statistical distances of DyDM compared to established models: DyDM learns the spectrum better in both the n-dimensional mean and the marginal Wasserstein sense, without requiring ad hoc data augmentation. Results are rounded to two decimal places, and results $(*)$ are equal until the fourth decimal, and $(\dagger)$ until the third decimal (see App. Q.6).

Dataset	WL-Bimodal		Comm. Small		Brain	
Distance	μ	$\mathcal{W}_{\text{marg}}$	μ	$\mathcal{W}_{\text{marg}}$	μ	$\mathcal{W}_{\text{marg}}$
DyDM (ours)	**0.02**	**0.01***	**0.07**	**0.02**	**0.05**	**0.03†**
EDP-GNN	0.13	0.08	0.42	0.14	0.07	0.03†
GDSS	0.24	0.13	0.38	0.14	0.33	0.12
ConGress	0.38	0.16	0.27	0.11	0.13	0.03†
DiGress (no trick)	1.06	0.29	2.51	0.45	0.57	0.17
DiGress (trick)	0.03	**0.01***	0.09	0.03	0.12	**0.03†**

Lemma 5 (Permutation invariance of score in Lie algebra)**.** *For any permutation matrix P we have* $\nabla \ln p_t(\bar{\lambda}, P\bar{X}) = \nabla \ln p_t(\bar{\lambda}, \bar{X}) \in \mathfrak{o}(n)$ *for all* $t \geq 0$, $\bar{\lambda} \in$ Weyl Chamber *and* $\bar{X} \in O(n)$.

Architecturally, this implies that a suitable architecture for learning the eigenvector information are Set Transformers [30], where one input set element corresponds to a row of X. We demonstrate in Figure 4 our numerical scheme for the time reversal on $SO(3)$.

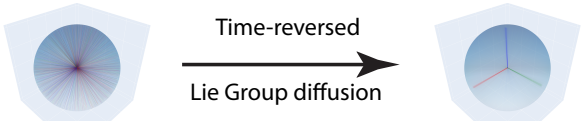


Figure 4: Numerical demonstration on $SO(3)$: Starting from the invariant distribution (left), a learned distribution is being generated (right) using Equation (7).

Scaling to larger dimensions. To scale to larger dimensions, we make the following observations. First, we exploit symmetry of eigenvectors under sign flips which do not influence the dynamics. This implies that the score obeys $\nabla \ln p_t(\bar{\lambda}, \bar{X}F) = F \nabla \ln p_t(\bar{\lambda}, \bar{X})F$ for any diagonal matrix F with entries ± 1, thus a reduction of the learning target of order 2^n, as we prove in App. S.6. Second, dropping the λ-dependent coefficients in the forward dynamics allows the SDE to retain its orthonormality guarantees and invariant state (for details, see Section S.7). We therefore implement for $n > 3$ an architecture that is invariant to sign flips – by identifying each Lie algebra increment with a canonical sign choice – and use the simplified Eigenvector SDE, i.e. without λ dependence in the update rule (note that the score still remains conditional on the spectrum).

4 Experiments

We empirically evaluate DyDM against several state-of-the-art graph diffusion models for learning graph spectra. First, we compare them on a simple bimodal distribution between two WL-equivalent graphs, illustrating the struggle of GNN-based methods (Section 4.1). Next, we carry out a comparison on a standard graph benchmark dataset and demonstrate scalability on a larger dataset (15,000 graphs).

We further demonstrate in App. S the accessibility of the eigenvectors by learning the Lie group SDE's time reversal of Section 3.3 on $SO(3)$ and with a numerical adjustment we learn a Lie group diffusion for the WL-Bimodal case.

4.1 Methodology

We compare to the GNN-based models EDP-GNN [8] and GDSS [9], as well as graph-transformer-based ConGress and DiGress [10], where among all the models only DiGress uses a data-augmentation trick: it adds certain graph features, including cycle counts and the first 6 eigenvalues (for details, see Section 5), but this *trick* is supposedly inessential for building a good model [10]. The trick, however, improves the learning of certain features, but not necessarily other subgraph structures [31]. We thus

Table 2: Comparison of graph statistics between DyDM and established graph generative models using maximum mean discrepancy (MMD) in 10^{-3} units on the brain dataset.

Model	Degree	Clustering	Orbit	Spectral	Mean
DyDM (ours)	**0.2**	**1.7**	**0.5**	**0.5**	**0.7** ($\pm$ 0.2)
EDP-GNN	1.2	2.2	1.4	1.1	1.5
GDSS	12.0	18.6	3.6	5.2	9.9
ConGress	1.3	2.2	2.0	1.1	1.7
DiGress (no trick)	2.0	6.3	13.9	11.5	8.4
DiGress (trick)	**0.9**	**1.9**	**1.2**	**0.7**	**1.2** ($\pm$ 0.4)

compare to both the DiGress model "without the trick", i.e., just a graph (Markov chain) diffusion model, and with the data-augmentation trick. As we evaluated the full spectrum while the published experimental results only showed partial or no information about the spectrum, we need explicit access to the samples. Since not all models [10, 8] report snapshots, we had to retrain those also on the standard datasets. We test DyDM on the following datasets, taking recent advice on limitations of certain graph benchmarks into account [32].

WL-Bimodal. We train the simple bimodal distribution in Fig. 2, consisting of 80% graph A and 20% graph B, which are WL equivalent. The dataset has $N = 5,000$ random permutations of A and B, and we follow the standard test/train split procedure [9, 33, 8] using 80% of the data as train data and the remaining 20% as test data. For hyperparameter tuning of the comparison models, see App. Q.

Community small. Being a standard benchmark [33, 8, 9], we include it for comparison. However, due to heavy undersampling, we can only test for memorization. From our perspective, memorization is the best that can be tested with said benchmark, and we elaborate on this in App. Q.1.

Brain. From the human connectome graph we drew $15,000$ ego-graphs (i.e. the induced subgraph of neighborhoods) of size 5 to 10 vertices [34, 35]. Crucially, this dataset demonstrates scalability to a number of graphs large enough for high statistical fidelity (faithful representation of the underlying distribution) in test and train set. For details, see App. P.

4.2 Results

In Table 1 we report mean distances and marginal Wasserstein distances of the spectra. Explicitly, if ν_{test} is the distribution of the spectrum of the test dataset and ν_{samp} the distribution of our samples, then the distance between the means in $\mathbb{R}^n$ is $\mu(\nu_{\text{samp}}, \nu_{\text{test}}) = \left\| \mathbb{E}_{\lambda\sim\nu_{\text{samp}}}[\lambda] - \mathbb{E}_{\lambda\sim\nu_{\text{test}}}[\lambda] \right\|_2$. For statistical feasibility, instead of calculating the full Wasserstein distance we use the averaged marginal Wasserstein distance given by $\mathcal{W}_{\text{marg}}(\nu_{\text{samp}}, \nu_{\text{test}}) = \frac{1}{n}\sum_{k=1}^{n} \mathcal{W}((\nu_{\text{samp}})_k, (\nu_{\text{test}})_k)$, for $(\nu_{\text{samp}})_k$ the marginal distribution in dimension k and $\mathcal{W}$ the Wasserstein distance between two 1d distributions. With these metrics we can evaluate both marginal and high-dimensional effects.

These metrics also reveal the limitations of GNN- and Graph-Transformer-based models on the simple WL-Bimodal dataset, as described in Figure 2. We also see that this extends to the real-world benchmark dataset Community Small. DyDM, on the other hand, consistently overcomes these issues. Even if we compare to the model with ad hoc feature augmentation (DiGress with "trick", on which we elaborate in Section 5), DyDM – which does not employ feature augmentation – either improves on or is on par with the feature augmented DiGress. We demonstrate that this performance still holds when working on large datasets, such as the $15,000$ ego-graphs from the Brain dataset.

With the learned eigenvectors we demonstrate through results for end-to-end graph generation in Table 2 for Brain and in Table 7 for WL-Bimodal that the remaining information remains indeed accessible and viable to generate accurate and valid eigenvectors and entire graphs.

5 Related Work

The spectrum carries key features of graphs [36], so that spectral methods have generally proven fruitful in graph research, exploiting information encoded in the spectrum, e.g. for graph comparison [37], or in dominant eigenvectors, such as in clustering, community detection [38, 39], and network embedding [40]. For graph diffusion models, however, the spectrum has been used only as either

auxiliary features for data augmentation [10], or in a way which makes any remaining $\Theta(n^2)$ degrees of freedom inaccessible, as we outline below.

Data augmentation. Ref. [10] acknowledges the importance of spectra of graphs and adds several auxiliary features, among them the first 6 eigenvalues of the Graph Laplacian as graph-level features and the first 2 non-zero eigenvectors as vertex-level features. While the "trick" of adding features is not principled, we show in Table 1 that these auxiliary features are *necessary* for DiGress to provide a good model. These features are not needed by DyDM, which builds on an *analytical* expression of the evolution of *all* eigenvalues during diffusion of a symmetric matrix.

Spectral methods. Spectral methods for graph learning have been employed in various contexts, for instance with GANs [41], which have, however, been outperformed by diffusion-based models [10]. Spectral information for graph *diffusion* models has been explored in recent work, however, none consider how diffusion of the spectrum impacts the remaining information (eigenvectors), leading to either a model that cannot guarantee eigenvector orthonormality [42], or a model *solely on the spectrum* as *if the eigenvectors were not impacted by diffusion* [17], hence making any recovery of remaining information in the form of eigenvectors impossible. Ref. [17] argues that the spectrum contains much information, so that upon sampling the spectrum from the OU-based model, the eigenvectors are simply sampled by taking the eigenvectors of a training sample chosen uniformly at random. However, this strategy is very limited, as it allows only for a diffusion in n parameters, losing $\Theta(n^2)$ degrees of freedom *irrecoverably*. Since DyDM learns the spectrum of an OU diffusion *on the entire graph*, the remaining information remains accessible, see Theorem 3. This allows learning the eigenvectors beyond sampling uniformly from training data [17]. Our Lie group formulation enforces orthonormality intrinsically, without unnecessarily overparameterizing the degrees of freedom (using $n(n-1)/2+n$ instead of $n(n+1)$) and without the need for rejection sampling [42].

6 Limitations and Extensions

Beyond graphs. Our experiments are limited to graphs. However, since the Dyson SDE is valid for the domain of $\mathrm{Sym}(\mathbb{R}^{n\times n})$, it is applicable beyond graphs. For instance, if correlations between n points are represented as covariance matrices, DyDM could learn the spectrum and thereby quantify the effective dimensionality or concentration of variance along principal components [43, 44, 45].

Larger graphs. Our experiments are performed on datasets of graphs with up to 20 nodes, i.e. $20 + \binom{20}{2} = 210$ degrees of freedom. Scaling our approach to very large graphs is a remaining extension.

Theoretical characterization. A theoretical extension could analyze the precise limitations of GNNs in a diffusion context, building on the WL classifications [14, 15].

Complex values. While our equations and numerics are limited to real-valued data, an interesting generalization of Dyson-BM is to the complex domain. Thus, generalizing to complex-weighted graphs [46, 47]: instead of working on $\mathrm{Sym}(\mathbb{R}^{n\times n})$, one works on the space of Hermitian matrices. Similarly, matrices over the algebra of real quaternions may be considered [24].

7 Conclusion

Leveraging the analytical insights offered by Dyson's Brownian motion from random matrix theory, we introduced DyDM, a diffusion model for spectral learning. Building on insights into the expressivity limitations of existing graph learning architectures, we decouple the dynamics so that the spectrum can be learned without permutation-equivariant networks. This approach avoids architecture-induced hallucination, enabling DyDM to learn the target distributions accurately and without data augmentation, as demonstrated experimentally. The eigenvector diffusion enables learning the remaining dynamics with the minimal number of degrees of freedom, allowing the neural network to learn the *distribution on the manifold* without needing to learn the manifold itself, thereby enabling full end-to-end graph generation. Furthermore, our framework lays the foundation for several future extensions with applications beyond graphs. We hope this work opens a new direction for enforcing inductive biases beyond the choice of architecture.

Acknowledgment We thank Jakiw Pidstrigach and Marc Roth for helpful discussions. Particular thanks are due to C. K. for hyperparameter tuning of and evaluation of the state-of-the-art models against which we compare. Financial support from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation program (Grant Agreement No. 101086182 to A. G.), the Rhodes Trust and EPSRC Centre for Doctoral Training in Mathematics of Random Systems: Analysis, Modelling and Simulation (EPSRC Grant EP/S023925/1) (to T. S.) and the Swiss National Science Foundation (to C.K. grant number 235409) are acknowledged. This work used the Scientific Compute Cluster at GWDG, the joint data center of Max Planck Society for the Advancement of Science (MPG) and University of Göttingen. In part funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 405797229. The authors acknowledge during the rebuttal time the use of resources provided by the Isambard-AI National AI Research Resource (AIRR). Isambard-AI [48] is operated by the University of Bristol and is funded by the UK Government's Department for Business, Innovation, Science and Trade (DBIST) via UK Research and Innovation; and the Science and Technology Facilities Council [ST/AIRR/I-A-I/1023]. Compute access was awarded through the UKRI AIRR Gateway route under project ID 0261-9981-8985-1.

Competing interests The authors declare no competing interests.

Acknowledgment of LLM use We acknowledge the help of LLMs to aid in implementing standard methods.

References

[1] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, 2015.

[2] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[4] Chenshuang Zhang, Chaoning Zhang, Sheng Zheng, Mengchun Zhang, Maryam Qamar, Sung-Ho Bae, and In So Kweon. A Survey on Audio Diffusion Models: Text To Speech Synthesis and Enhancement in Generative AI. *arXiv preprint arXiv:2303.13336*, 2023.

[5] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion Models in Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, September 2023.

[6] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, August 2023.

[7] Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization. In *Advances in Neural Information Processing Systems*, 2023.

[8] 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.

[9] 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.

[10] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete Denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023.

[11] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, 2016.

[12] Han Huang, Leilei Sun, Bowen Du, Yanjie Fu, and Weifeng Lv. GraphGDP: Generative Diffusion Processes for Permutation Invariant Graph Generation. In *IEEE International Conference on Data Mining*, 2022.

[13] 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. In *Advances in Neural Information Processing Systems*, 2024.

[14] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *AAAI Conference on Artificial Intelligence*, 2019.

[15] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

[16] Jiaoyang Huang and Horng-Tzer Yau. Spectrum of random d-regular graphs up to the edge. *Communications on Pure and Applied Mathematics*, 77(3):1635–1723, 2024. ISSN 1097-0312.

[17] Tianze Luo, Zhanfeng Mo, and Sinno Jialin Pan. Fast Graph Generation via Spectral Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:3496–3508, 2024.

[18] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning: With Applications in Python*. Springer Texts in Statistics. Springer, 2023.

[19] Grigorios A. Pavliotis. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*. Texts in Applied Mathematics. Springer New York, 2014.

[20] Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[21] Markus Meringer. Fast generation of regular graphs and construction of cages. *Journal of Graph Theory*, 30(2):137–146, 1999.

[22] Monique M. Tirion. Large Amplitude Elastic Motions in Proteins from a Single-Parameter, Atomic Analysis. *Physical Review Letters*, 77(9):1905–1908, 1996.

[23] Turkan Haliloglu, Ivet Bahar, and Burak Erman. Gaussian Dynamics of Folded Proteins. *Physical Review Letters*, 79(16):3090–3093, October 1997.

[24] Freeman J. Dyson. A Brownian-Motion Model for the Eigenvalues of a Random Matrix. *Journal of Mathematical Physics*, 3(6):1191–1198, 1962.

[25] Romain Allez, Joël Bun, and Jean-Philippe Bouchaud. The eigenvectors of Gaussian matrices with an external source. *arXiv preprint arXiv:1412.7108*, 2014.

[26] Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian Score-Based Generative Modelling. *Advances in Neural Information Processing Systems*, 35:2406–2422, December 2022.

[27] Fang Yang. Topics in random matrix theory. https://yangf75.github.io/RMT(2022Fall,49-61).pdf, 2022.

[28] Marco Bertolini, Tuan Le, and Djork-Arné Clevert. Diffusion Generative Modeling on Lie Group Representations. In *Advances in Neural Information Processing Systems*, October 2025.

[29] Goran Marjanovic, Marc J. Piggott, and Victor Solo. Numerical methods for stochastic differential equations in the Stiefel manifold made simple. In *IEEE 55th Conference on Decision and Control (CDC)*, 2016.

[30] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, 2019.

[31] Xiyuan Wang, Yewei Liu, Lexi Pang, Siwei Chen, and Muhan Zhang. Do Graph Diffusion Models Accurately Capture and Generate Substructure Distributions? *arXiv preprint arXiv:2502.02488*, 2025.

[32] Maya Bechler-Speicher, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M. Bronstein, Mathias Niepert, Bryan Perozzi, Mikhail Galkin, and Christopher Morris. Position: Graph Learning Will Lose Relevance Due To Poor Benchmarks. In *International Conference on Machine Learning Position Paper Track*, 2025.

[33] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *International Conference on Machine Learning*, 2018.

[34] Katrin Amunts, Claude Lepage, Louis Borgeat, Hartmut Mohlberg, Timo Dickscheid, Marc-Étienne Rousseau, Sebastian Bludau, Pierre-Louis Bazin, Lindsay B. Lewis, Ana-Maria Oros-Peusquens, Nadim J. Shah, Thomas Lippert, Karl Zilles, and Alan C. Evans. BigBrain: An Ultrahigh-Resolution 3D Human Brain Model. *Science*, 340(6139):1472–1475, 2013.

[35] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI Conference on Artificial Intelligence*, 2015. URL `http://networkrepository.com`.

[36] Andries E. Brouwer and Willem H. Haemers. *Spectra of Graphs*. Universitext. Springer New York, New York, NY, 2012.

[37] Richard C Wilson and Ping Zhu. A study of graph spectra for comparing graphs and trees. *Pattern Recognition*, 41(9):2833–2841, 2008.

[38] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[39] Mark EJ Newman. Spectral methods for community detection and graph partitioning. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 88(4):042822, 2013.

[40] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[41] 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*, 2022.

[42] Giorgia Minello, Alessandro Bicciato, Luca Rossi, Andrea Torsello, and Luca Cosmo. Generating graphs via spectral diffusion. In *International Conference on Learning Representations*, 2025.

[43] H. Y. Chen, Raphaël Liégeois, John R. de Bruyn, and Andrea Soddu. Principal-component analysis of particle motion. *Physical Review E*, 91(4):042308, 2015.

[44] Maxime Estavoyer and Olivier François. Theoretical analysis of principal components in an umbrella model of intraspecific evolution. *Theoretical Population Biology*, 148:11–21, December 2022.

[45] Berk Hess. Similarities between principal components of protein dynamics and random diffusion. *Physical Review E*, 62(6):8438–8448, December 2000.

[46] Yu Tian and Renaud Lambiotte. Structural Balance and Random Walks on Complex Networks with Complex Weights. *SIAM Journal on Mathematics of Data Science*, 6(2):372–399, June 2024.

[47] Cristina López Amado, Tassilo Schwarz, Yu Tian, and Renaud Lambiotte. Complex-Weighted Convolutional Networks: Provable Expressiveness via Complex Diffusion. In *The Fourth Learning on Graphs Conference*, October 2025.

[48] Simon McIntosh-Smith, Sadaf R. Alam, and Christopher Woods. Isambard-AI: A leadership class supercomputer optimised specifically for Artificial Intelligence, November 2024.

[49] John P. Keating. Random Matrix Theory. Lecture Notes, Oxford University, 2023.

[50] Greg W. Anderson, Alice Guionnet, and Ofer Zeitouni. *An Introduction to Random Matrices*. Cambridge University Press, 2009.

[51] Makoto Katori and Hideki Tanemura. Noncolliding Brownian motions and Harish-Chandra formula. *Electronic Communications in Probability*, 8:112–121, 2003.

[52] Bernt Øksendal. *Stochastic Differential Equations*. Universitext. Springer Berlin Heidelberg, 2003.

[53] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and Improving the Training Dynamics of Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[54] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, 2020.

[55] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[56] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

[57] Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems*, 2021.

[58] Elton P. Hsu. *Stochastic Analysis on Manifolds*, volume 38 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2002.

[59] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2018.

Appendix

The Appendix is structured as follows. We first derive the tractable loss $\tilde{L}(\theta)$ in Section A. We then provide proofs of the (Dyson-BM) SDE in Section B as well as for the (Eigenvector-SDE) in Section C. We then explain the applicability of Anderson's time reversal (Section D), and provide a time-rescaling of (Dyson-BM) in Section E. In Section F, we provide the adaptive step size controller, followed by an explanation of the shooting mechanism (Section G). We provide a theoretical analysis of the complexity of the numerical update steps for (Dyson-BM) and (Eigenvector-SDE) in dependence of the graph size n in Section H, followed by a discussion of empirical resource considerations in Section I. We then explain the sampling procedure (Section J) and give engineering details (Section K). We explain the challenges of Dyson's Brownian motion which we overcame with DyDM in Section L. We present our theoretical argument for using inductive bias in Section M and prove the Lemma on WL equivalence of k-regular graphs in Section N. In Section O, we demonstrate DyDM on the Graph Laplacian, showing its applicability beyond adjacency spectra and show how it can learn properties such as the algebraic connectivity (Fiedler value). For the experiments, we first elaborate on the datasets (Section P) followed by an explanation of our extensive benchmarking in Section Q, including a comment about undersampling in some benchmark datasets in Section Q.1. We show the learning dynamics of 4 different runs of EDP-GNN in Section R. We give mathematical details for the extension to (Eigenvector-SDE) and its time-reversal in Section S, and demonstrate the recoverability of the remaining information in Table 7 and by plotting the generated graphs in Fig. 12.

A Making the loss tractable

In this section, we deduce a general loss formula for an SDE on $\mathbb{R}$ that will be applied to (Dyson-BM). We consider the process $X = (X(t))_{t\geq 0}$ determined by the SDE

$$\mathrm{d}X(t) = a(t, X(t))\mathrm{d}t + b(t, X(t))\mathrm{d}W_t$$

with initial condition $X(0)$. We assume throughout this section that $X(t)$ is absolutely continuous for all $t \geq 0$ and denote by p_t the density of $X(t)$. We furthermore assume that the joint densities of $X(t)$ and $X(s)$ also have density for all $t, s \geq 0$ that we write as $p_{t,s}$. Observe that by Bayes formula for $t \geq s \geq 0$ we have for $x, y \in \mathbb{R}^d$ that

$$p_{t,s}(y, x) = p_{t|s}(y|x)p_s(x), \tag{8}$$

where $p_{t|s}(y|x)$ is the conditional density of y given x.

The canonical loss arising from $s(y, t) = \boldsymbol{\nabla}_{M(t)} \ln p_t(M)$ is

$$L'(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,T], M(0)\sim p_0, M(t)\sim p_{t|0}(\cdot|M(0))}\left[\left\|s_\theta(M(t), t) - \boldsymbol{\nabla}_{M(t)} \ln p_t(M(t))\right\|_2^2\right]. \tag{9}$$

Note that the difference between $L'(\theta)$ and $L(\theta)$ is that we use the gradient of the *conditional* density $\boldsymbol{\nabla}_{M(t)} \ln p_{t|0}(M(t)|M(0))$ in $L(\theta)$. It is a well-known fact that $L'(\theta) = L(\theta) + \mathrm{const}(\theta)$. We will first explain how the loss $L'(\theta)$ from (9) and $\tilde{L}(\theta)$ from (4) are the same up to a constant, that is $L'(\theta) = \tilde{L}(\theta) + \mathrm{const}(\theta)$, which therefore results in the same gradient descent as with $L(\theta)$.

We generalize the loss from (9) to weighting time by a function η. So we consider the following generalized loss

$$\begin{aligned} L'(\theta) \coloneqq& \frac{1}{T}\int_0^T \eta(t) \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} \|s_\theta(y, t) - \boldsymbol{\nabla}_y \ln(p_t(y))\|_2^2 \, p_{t,0}(y, x)\,\mathrm{d}y\mathrm{d}x\mathrm{d}t \\ =& \frac{1}{T}\int_0^T \eta(t) \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} \|s_\theta(y, t) - \boldsymbol{\nabla}_y \ln(p_t(y))\|_2^2 \, p_{t|0}(y \mid x)\mathrm{d}y p_0(x)\mathrm{d}x\mathrm{d}t, \end{aligned} \tag{10}$$

where $\eta(t)$ is some weighting function with $\int_0^T \eta(t)\mathrm{d}t = 1$. We wrote it in the above form with $p_{t|0}$ denoting the conditional density, since the three integrals can be replaced by $\mathbb{E}_{t\sim U[0,T]}\left[\cdots \mathbb{E}_{x\sim p_0}\left[\mathbb{E}_{y\sim p_{t|0}(\cdot|x)}\left[\|\cdots\|_2^2\right]\right]\right]$, which we can sample from if we assume (1) sample access to p_0, (2) known density at any t given a dirac-delta p_0, (3) the term in the norm is tractable. (1) is assumed by the problem definition, (2) is known for an Ornstein-Uhlenbeck forward SDE, (3)

can be solved by realizing that there is an equivalent loss function $\hat{L}(\theta) = L(\theta) + \text{const}(\theta)$ where p_t inside the norm is converted to a $p_{t|0}$, which is known (Gaussian density) in an OU setting.

Here, steps (2) and (3) fail. Note that by the polarization identity

$$\|s_\theta(y,t) - \boldsymbol{\nabla}_y \ln(p_t(y))\|_2^2 = \|s_\theta(y,t)\|_2^2 + \|\boldsymbol{\nabla}_y \ln(p_t(y))\|_2^2 - 2s_\theta(y,t)^T \boldsymbol{\nabla}_y \ln(p_t(y))$$

We will rewrite the mixed term $s_\theta(y,t)^T \boldsymbol{\nabla}_y \ln(p_t(y))$ of the L^2 norm in such a way that the first quadratic term $\|s_\theta(y,t)\|_2^2$ remains unchanged and the second quadratic term $\|\boldsymbol{\nabla}_y \ln(p_t(y))\|_2^2$ is constant in θ. First, we rewrite

$$\begin{aligned}
&\frac{1}{T}\int_0^T \eta(t) \int_{\mathbb{R}^n \times \mathbb{R}^n} s_\theta(y,t)^T \boldsymbol{\nabla}_y \ln(p_t(y)) p_{t,0}(y,x)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t \qquad (11)\\
=&\frac{1}{T}\int_0^T \eta(t) \int_{\mathbb{R}^n} s_\theta(y,t)^T \boldsymbol{\nabla}_y \ln(p_t(y)) p_t(y)\,\mathrm{d}y\,\mathrm{d}t\\
=&\frac{1}{T}\int_0^T \eta(t) \int_{\mathbb{R}^n} s_\theta(y,t)^T \boldsymbol{\nabla}_y [p_t(y)]\,\mathrm{d}y\,\mathrm{d}t. \qquad (12)
\end{aligned}$$

We next observe that for $0 \le s' \le s$ we have the following:

$$\begin{aligned}
\boldsymbol{\nabla}_y [p_s(y)] &= \boldsymbol{\nabla}_y \left[\int_{\mathbb{R}^n} p_{s|s'}(y|z) p_{s'}(z)\,\mathrm{d}z\right]\\
&= \int_{\mathbb{R}^n} \boldsymbol{\nabla}_y p_{s|s'}(y|z) p_{s'}(z)\,\mathrm{d}z\\
&= \int_{\mathbb{R}^n} \boldsymbol{\nabla}_y p_{s|s'}(y|z) \frac{p_{s,s'}(y,z)}{p_{s|s'}(y|z)}\,\mathrm{d}z\\
&= \int_{\mathbb{R}^n} \boldsymbol{\nabla}_y \ln p_{s|s'}(y|z) p_{s,s'}(y,z)\,\mathrm{d}z.
\end{aligned}$$

So we now perform in (12) for a small $h > 0$ a change of variables to $t + h$ and apply the latter equality with $s = t + h$ and $s' = t$. Then up to ignoring the boundary at 0 and T, and using by (8) that $p_{t+h,t}(y,z) = p_{t+h|t}(y|z)p_t(z)$ it follows that the loss from (4)

$$(12) = \frac{1}{T}\int_0^T \eta(t+h) \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} s_\theta(y,t+h)^T \boldsymbol{\nabla}_y \ln p_{t+h|t}(y|z) p_{t+h|t}(y|z)\,\mathrm{d}y\, p_t(z)\,\mathrm{d}z\,\mathrm{d}t. \quad (13)$$

So it follows that

$$\tilde{L}(\theta) = \frac{1}{T}\int_0^T \eta(t+h) \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} \big\|s_\theta(y,t+h) - \boldsymbol{\nabla}_y \ln\big(p_{t+h|t}(y|z)\big)\big\|_2^2 p_{t+h|t}(y|z)\,\mathrm{d}y\, p_t(z)\,\mathrm{d}z\,\mathrm{d}t \quad (14)$$

is equal to $L(\theta)$ up to a constant term in θ (ignoring the boundary terms at 0 and T).

We will now make a series of approximations to calculate the loss $\tilde{L}(\theta)$. The first one is to approximate the integral $\int_0^T$ over t by a sum $\sum_{i=1}^k$ over the time points $t_0 < t_1 < \ldots < t_k$ with $t_0 = 0$ and $t_k = T$. The second approximation we make is that we approximate the latter integral $\int_{\mathbb{R}^n}\int_{\mathbb{R}^n}$ by sampling a path from p_t at the time steps t_i. Indeed, we denote for each integer $1 \le r \le N$ by $x_{t_i}^{(r)}$ the sample path of p_t. Moreover, we actually make the time grid also dependent on our sample path. So for each $1 \le r \le N$, let $t_0^{(r)} < t_1^{(r)} < \ldots < t_{k^{(r)}}^{(r)}$ with $t_0^{(r)} = 0$ and $t_{k^{(r)}}^{(r)} = T$ be the discretization of $[0,T]$. Thus, the overall loss can be approximated as

$$\begin{aligned}
\tilde{L}(\theta) \approx &\frac{1}{N}\sum_{r=1}^N \sum_{i=1}^{k^{(r)}} \frac{t_i^{(r)} - t_{i-1}^{(r)}}{T} \eta(t_i^{(r)}) \Big\| s_\theta(x_i^{(r)}, t_i^{(r)}) - \boldsymbol{\nabla}_{x_i^{(r)}} \left[\ln\Big(p_{t_i^{(r)}|t_{i-1}^{(r)}}(x_i^{(r)} \mid x_{i-1}^{(r)})\Big)\right]\Big\|_2^2\\
&+ \text{const}(\theta). \qquad (15)
\end{aligned}$$

We finally approximate the incremental score function $\boldsymbol{\nabla}_{x_i^{(r)}} \left[\ln\Big(p_{t_i^{(r)}|t_{i-1}^{(r)}}(x_i^{(r)} \mid x_{i-1}^{(r)})\Big)\right]$ as follows. If $h > 0$ is a small time step, we can approximate the conditional random variable $X_{t+h} \mid \mathcal{F}_t$ with

$x := X_t$ by

$$X_{t+h} \mid \mathcal{F}_t \approx x + a(t,x)h + b(t,x)N(0,h)$$
$$\text{(where } N(0,h) \text{ is a centered Gaussian RV with variance } h)$$
$$\sim \mathcal{N}\left(x + a(t,x)h,\, b^2(t,x)h\right), \tag{16}$$

so that we have for the density

$$p_{t+h|t}(y|x) \approx \frac{1}{\sqrt{2\pi b^2(t,x)h}} \exp\left(-\frac{(y-x-a(t,x)h)^2}{2b^2(t,x)h}\right), \tag{17}$$

which means for the scores

$$\boldsymbol{\nabla}_y \ln\left(p_{t+h|t}(y|x)\right) \approx -\frac{y-x-a(t,x)h}{b^2(t,x)h}. \tag{18}$$

Thus, combining (15) and (18), the loss $\tilde{L}(\theta)$ can be approximated by the following as we use in our model:

$$\frac{1}{N}\sum_{r=1}^{N}\sum_{i=1}^{k^{(r)}} \frac{t_i^{(r)} - t_{i-1}^{(r)}}{T}\eta(t_i^{(r)}) \left\| s_\theta(x_i^{(r)}, t_i^{(r)}) - \frac{a(t_{i-1}^{(r)}, x_{i-1}^{(r)}) \cdot (t_i^{(r)} - t_{i-1}^{(r)}) - (x_i^{(r)} - x_{i-1}^{(r)})}{b^2(t_{i-1}^{(r)}, x_{i-1}^{(r)}) \cdot (t_i^{(r)} - t_{i-1}^{(r)})} \right\|_2^2 \tag{19}$$

B Spectral Dyson SDE

Theorem 2 (restated). *Denote by $\lambda(t) = (\lambda_1(t), \ldots, \lambda_n(t))$ the ordered spectrum of the matrix-valued Ornstein-Uhlenbeck process $M(t)$ of Equation* (1)*. Then assuming that the initial matrix $M(0)$ has simple spectrum, $\lambda(t)$ satisfies for all $1 \le k \le n$ the stochastic differential equation*

$$\mathrm{d}\lambda_k(t) = \left(\alpha \sum_{\ell \neq k} \frac{1}{\lambda_k(t) - \lambda_\ell(t)} - \beta\lambda_k(t)\right)\mathrm{d}t + \sqrt{2\alpha}\mathrm{d}W_k(t), \tag{Dyson-BM}$$

for $W_1, \ldots, W_n$ independent standard Brownian motions. Moreover the unique stationary distribution of (Dyson-BM) *has density $p_{\mathrm{inv}}(\lambda) = \exp(-U(\lambda))/Z$ for $U(\lambda) = \frac{\beta}{2\alpha}\sum_k \lambda_k^2 - \sum_{k<\ell} \ln|\lambda_k - \lambda_\ell|$, $\lambda \in C_n$, and Z a normalizing constant so that p_{inv} corresponds to a probability measure.*

Proof. We prove the theorem in two parts.

Dyson SDE. We first prove (Dyson-BM). This part of the proof is based on [49] but generalizes it to arbitrary coefficients. For mathematical details on the $\alpha = \frac{1}{n}$, $\beta = 0$ case, see also [50]. Suppose $M(t)$ satisfies the SDE Equation (1). That is, with $M(0) \in \mathrm{Sym}(\mathbb{R}^{n\times n})$ we have

$$\mathrm{d}M_{ij}(t) = -\beta M_{ij}\mathrm{d}t + D_{ij}\mathrm{d}B_{ij}$$

with $D_{ij} := \sqrt{(1+\delta_{ij})\alpha}$ for any constants $\alpha, \beta \in \mathbb{R}^+$ with $B_{ij}(t) = B_{ji}(t)$ $\forall t > 0$. Let $\lambda_1 \ge \ldots \ge \lambda_n$ be the eigenvalues. We choose as $v_1, \ldots, v_n$ an orthonormal basis of eigenvectors ($Mv_k = \lambda_k v_k$), which exists by the spectral theorem for symmetric matrices.

Due to symmetry, we may constrain the set of indices (i,j) to $\mathcal{I} := \{(i,j) \mid 1 \le i \le j \le n\}$. For any $k \in [n]$, the eigenvalue λ_k can thus be seen as a function of the set of Itô processes $\{M_\eta \mid \eta \in \mathcal{I}\}$. Hence, we have by Itô's Lemma,

$$\mathrm{d}\lambda_k = \underbrace{\frac{\partial\lambda_k}{\partial t}}_{\equiv 0}\mathrm{d}t + \sum_{\eta\in\mathcal{I}} \frac{\partial\lambda_k}{\partial M_\eta}\mathrm{d}M_\eta + \frac{1}{2}\sum_{\eta,\xi\in\mathcal{I}} \frac{(\partial\lambda_k)^2}{\partial M_\eta \partial M_\xi}\mathrm{d}M_\eta \mathrm{d}M_\xi, \tag{20}$$

where the first part is 0 since $\lambda_k(t)$ is only a function of the $M_\eta(t)$, not of time. By Equation (1), we get

$$= \sum_{\eta\in\mathcal{I}} \left(-\beta M_\eta \frac{\partial\lambda_k}{\partial M_\eta} + \frac{1}{2}D_\eta^2 \frac{\partial^2\lambda_k}{(\partial M_\eta)^2}\right)\mathrm{d}t + D_\eta \frac{\partial\lambda_k}{\partial M_\eta}\mathrm{d}B_\eta. \tag{21}$$

It remains to calculate the partial derivatives. In what follows, we successively apply properties of the spectrum in order to obtain equations for the partial derivatives. We have

$$Mv_k = \lambda_k v_k \tag{22}$$

taking the partial derivative with respect to M_{ij} for $(i,j) \in \mathcal{I}$ on both sides and applying the product rule yields

$$\frac{\partial M}{\partial M_{ij}} v_k + M \frac{\partial v_k}{\partial M_{ij}} = \frac{\partial \lambda_k}{\partial M_{ij}} v_k + \lambda_k \frac{\partial v_k}{\partial M_{ij}}. \tag{23}$$

Further, by orthogonality of the eigenvectors we have for any $k, l \in [n]$

$$v_k^T v_l = \delta_{kl} \tag{24}$$

$$\frac{\partial v_k^T}{\partial M_{ij}} v_l + v_k^T \frac{\partial v_l}{\partial M_{ij}} = 0. \tag{25}$$

which, taking the transpose, implies for $l = k$

$$\frac{\partial v_k^T}{\partial M_{ij}} v_k = v_k^T \frac{\partial v_k}{\partial M_{ij}} = 0. \tag{26}$$

Multiplying Equation (23) by v_l^T from the left yields

$$v_l^T \frac{\partial M}{\partial M_{ij}} v_k + \underbrace{v_l^T M \frac{\partial v_k}{\partial M_{ij}}}_{(I)} = \underbrace{\frac{\partial \lambda_k}{\partial M_{ij}} v_l^T v_k}_{(II)} + \underbrace{\lambda_k v_l^T \frac{\partial v_k}{\partial M_{ij}}}_{(III)} \tag{27}$$

where the terms simplify as follows.

$$(I) = (1 - \delta_{kl}) \lambda_l v_l^T \frac{\partial v_k}{\partial M_{ij}} \qquad \text{(by Equation (26))}$$

$$(II) = \delta_{lk} \frac{\partial \lambda_k}{\partial M_{ij}} \tag{28}$$

$$(III) = (1 - \delta_{kl}) \lambda_k v_l^T \frac{\partial v_k}{\partial M_{ij}}, \qquad \text{(by Equation (26))}$$

resulting in the following equation for $l = k$

$$v_k^T \frac{\partial M}{\partial M_{ij}} v_k = \frac{\partial \lambda_k}{\partial M_{ij}} \tag{29}$$

and for $l \neq k$:

$$v_l^T \frac{\partial M}{\partial M_{ij}} v_k + \lambda_l v_l^T \frac{\partial v_k}{\partial M_{ij}} = \lambda_k v_l^T \frac{\partial v_k}{\partial M_{ij}} \tag{30}$$

$$v_l^T \frac{\partial M}{\partial M_{ij}} v_k = (\lambda_k - \lambda_l) \, v_l^T \frac{\partial v_k}{\partial M_{ij}}. \tag{31}$$

The matrix derivative is, trivially,

$$\left(\frac{\partial M}{\partial M_{ij}} \right)_{kl} = \begin{cases} 1 & \text{for } (k,l) = (i,j) \text{ or } (l,k) = (i,j) \\ 0 & \text{else} \end{cases}, \tag{32}$$

so that we can simplify Equation (29) to

$$\frac{\partial \lambda_k}{\partial M_{ij}} = (v_k)_i (v_k)_j (2 - \delta_{ij}) \tag{33}$$

and Equation (31) to

$$(\lambda_k - \lambda_l) \, v_l^T \frac{\partial v_k}{\partial M_{ij}} = (v_k)_i (v_l)_j + (1 - \delta_{ij}) (v_k)_j (v_l)_i \tag{34}$$

$$= (v_l)_i (v_k)_j + (1 - \delta_{ij}) (v_l)_j (v_k)_i. \tag{35}$$

Using that the v_k are an ONB of eigenvectors, we can conclude with Equation (33) that the first summand of Equation (21) is

$$\sum_{\eta \in \mathcal{I}} -\beta M_\eta \frac{\partial \lambda_k}{\partial M_\eta} = -\beta \lambda_k. \tag{36}$$

Since $\{v_1, \dots, v_n\}$ is an orthonormal basis, we may project on the basis vectors as follows

$$\frac{\partial v_k}{\partial M_{ij}} = \sum_{l \in [n]} v_l^T \frac{\partial v_k}{\partial M_{ij}} v_l \qquad \text{(projection into ONB basis)}$$

applying Equation (26) gives

$$= \sum_{l \in [n] \setminus \{k\}} v_l^T \frac{\partial v_k}{\partial M_{ij}} v_l \tag{37}$$

which yields with Equation (34)

$$= \sum_{l \in [n] \setminus \{k\}} \frac{1}{\lambda_k - \lambda_l} \left((v_k)_i (v_l)_j + (1 - \delta_{ij}) (v_k)_j (v_l)_i \right) v_l. \tag{38}$$

To obtain the second order partial derivative of λ_k, we observe

$$\begin{aligned}
\frac{\partial^2 \lambda_k}{\partial M_{ij} \partial M_{ij}} =& \frac{\partial}{\partial M_{ij}} \left((v_k)_i (v_k)_j (2 - \delta_{ij}) \right) && \text{(by Equation (33))} \\
=& (2 - \delta_{ij}) \left(\frac{\partial (v_k)_i}{\partial M_{ij}} (v_k)_j + (v_k)_i \frac{\partial (v_k)_j}{\partial M_{ij}} \right) \\
=& (2 - \delta_{ij}) \sum_{l \in [n] \setminus \{k\}} \frac{1}{\lambda_k - \lambda_l} \Big((v_l)_i (v_k)_j (v_l)_i (v_k)_j + (v_l)_j (v_k)_i (1 - \delta_{ij}) (v_l)_i (v_k)_j \\
& + (v_l)_i (v_k)_j (v_l)_j (v_k)_i + (v_l)_j (v_k)_i (1 - \delta_{ij}) (v_l)_j (v_k)_i \Big) && \text{(by Equation (38))} \\
=& (2 - \delta_{ij}) \sum_{l \in [n] \setminus \{k\}} \frac{1}{\lambda_k - \lambda_l} \Big((v_l)_i^2 (v_k)_j^2 + (v_l)_j^2 (v_k)_i^2 (1 - \delta_{ij}) \\
& + (v_l)_i (v_l)_j (v_k)_i (v_k)_j (2 - \delta_{ij}) \Big). && (39)
\end{aligned}$$

These second order partial derivative are summed over $\mathcal{I}$ in Equation (21), so that combining Equation (39) with the definition of D_{ij} yields

$$\begin{aligned}
\sum_{(i,j) \in \mathcal{I}} D_{ij}^2 \frac{\partial^2 \lambda_k}{(\partial M_{ij})^2} =& \sum_{j=1}^{n} \sum_{i=1}^{j} \alpha (1 + \delta_{ij})(2 - \delta_{ij}) \sum_{l \in [n] \setminus \{k\}} \frac{1}{\lambda_k - \lambda_l} \Big((v_l)_i^2 (v_k)_j^2 \\
& + (v_l)_j^2 (v_k)_i^2 (1 - \delta_{ij}) + (v_l)_i (v_l)_j (v_k)_i (v_k)_j (2 - \delta_{ij}) \Big) && (40)
\end{aligned}$$

we reorder and note that since the summand is symmetric in i, j, we may change the range of summation and absorb the coefficient $2 - \delta_{ij}$,

$$\begin{aligned}
=& \alpha \sum_{l \in [n] \setminus \{k\}} \frac{1}{\lambda_k - \lambda_l} \sum_{j=1}^{n} \sum_{i=1}^{n} (1 + \delta_{ij}) \Big((v_l)_i^2 (v_k)_j^2 \\
& + (v_l)_j^2 (v_k)_i^2 (1 - \delta_{ij}) + (v_l)_i (v_l)_j (v_k)_i (v_k)_j (2 - \delta_{ij}) \Big) && (41)
\end{aligned}$$

where we realize that upon accounting for all δ_{ij}, the first two summands are simply $\|v_l\|_2^2\|v_k\|_2^2$, while the third summand may be written an inner product

$$= \alpha \sum_{l\in[n]\setminus\{k\}} \frac{1}{\lambda_k - \lambda_l} \left(2\|v_l\|_2^2\|v_k\|_2^2 + 2(\underbrace{v_l^T v_k}_{0})^2 \right) \tag{42}$$

since we chose an ONB, this simplifies to

$$= 2\alpha \sum_{l\in[n]\setminus\{k\}} \frac{1}{\lambda_k - \lambda_l}. \tag{43}$$

It remains to determine an explicit expression of the Brownian motion in Equation (21). We have

$$\sum_{\eta\in\mathcal{I}} D_\eta \frac{\partial \lambda_k}{\partial M_\eta} \mathrm{d}B_\eta = \sum_{(i,j)\in\mathcal{I}} \sqrt{(1+\delta_{ij})\alpha}(2-\delta_{ij})(v_k)_i(v_k)_j \mathrm{d}B_{ij}$$

using that $B_{ij}(t) = B_{ji}(t)$, we obtain

$$= \sqrt{2\alpha} \sum_{i=1}^{n}\sum_{j=1}^{n} \sqrt{\frac{1+\delta_{ij}}{2}}(v_k)_i(v_k)_j \mathrm{d}B_{ij}$$

we may now define $\mathrm{d}\tilde{B}_k$ as follows

$$= \sqrt{2\alpha}\mathrm{d}\tilde{B}_k. \tag{44}$$

Indeed, the set $\{\tilde{B}_k \mid k \in [n]\}$ is in distribution equal n independent standard Brownian motions, since $\mathbb{E}\left[\mathrm{d}\tilde{B}_k\right] = 0$ and for $k, l \in [n]$

$$\mathbb{E}\left[\mathrm{d}\tilde{B}_k \mathrm{d}\tilde{B}_l\right] = \mathbb{E}\left[\frac{1}{2}\sum_{ij}\sum_{st}\sqrt{1+\delta_{ij}}\sqrt{1+\delta_{st}}(v_k)_i(v_k)_j(v_l)_s(v_l)_t \mathrm{d}B_{ij}\mathrm{d}B_{st}\right]$$

where we realize that the product of the differentials is 0 except for $(i,j) = (s,t)$ and $(i,j) = (t,s)$

$$\begin{aligned} &= \mathbb{E}\left[v_k^T v_l v_k^T v_l\right] \mathrm{d}t \\ &= \delta_{kl}\mathrm{d}t. \end{aligned} \tag{45}$$

We can thus conclude that by Equation (21) we have,

$$\mathrm{d}\lambda_k = \left(-\beta\lambda_k + \alpha \sum_{l\in[n]\setminus\{k\}} \frac{1}{\lambda_k - \lambda_l} \right) \mathrm{d}t + \sqrt{2\alpha}\mathrm{d}W_k \tag{46}$$

where $\{W_k \mid k \in [n]\}$ are n independent Brownian motions.

Invariant Distribution. We now show that

$$p_{\mathrm{inv}}(\lambda) = \frac{1}{Z}\exp(-U(\lambda)) \qquad \text{for} \qquad U(\lambda) = \frac{\beta}{2\alpha}\sum_k \lambda_k^2 - \sum_{k<\ell} \ln|\lambda_k - \lambda_\ell|, \tag{47}$$

for $\lambda \in C_n$ and Z a normalizing constant so that p_{inv} is a probability measure, is the invariant distribution. We use the Fokker-Planck equation. Indeed, recall that if we have an SDE

$$\mathrm{d}\lambda_t = f(\lambda_t)\mathrm{d}t + L(\lambda_t)\mathrm{d}B_t$$

in dimension n, where $f(\lambda) = (f_1(\lambda), \ldots, f_n(\lambda))$ is a C^2-function, $L(\lambda)$ is matrix-valued C^2-function and B_t is a n-dimensional Brownian motion, then an invariant distribution $p(x)$ satisfies

$$\sum_{i=1}^{n} \frac{\partial}{\partial \lambda_i}[f_i(\lambda)p(\lambda)] = \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2}{\partial \lambda_i \partial \lambda_j}[L(\lambda)L(\lambda)^T]_{ij}p(\lambda). \tag{48}$$

In our case, for the $\lambda(\alpha, \beta)$-SDE we have that $L(\lambda, t) = \sqrt{2\alpha}1_n$ and therefore $[L(\lambda, t)L(\lambda, t)^T]_{ij} = 2\alpha\delta_{ij}$. So the right hand side of (48) equals

$$\alpha \sum_{i=1}^{n} \frac{\partial^2}{(\partial \lambda_i)^2} p(\lambda).$$

Assume for now that $\lambda = (\lambda_1, \ldots, \lambda_n)$ satisfies $\lambda_1 > \lambda_2 > \ldots > \lambda_n$ and

$$U(\lambda) = c\sum_i \lambda_i^2 - \sum_{i<j} \ln(\lambda_i - \lambda_j)$$

for some constant $c > 0$ to be determined. We first calculate for a fixed i,

$$\begin{aligned}
\frac{\partial}{\partial \lambda_i} p(\lambda) &= -\frac{1}{Z}\exp(-U(\lambda))\frac{\partial}{\partial \lambda_i}U(\lambda) \\
&= -\frac{1}{Z}\exp(-U(\lambda))\left(2c\lambda_i - \sum_{i<j}\frac{1}{\lambda_i - \lambda_j} + \sum_{j<i}\frac{1}{\lambda_j - \lambda_i}\right) \\
&= -\frac{1}{Z}\exp(-U(\lambda))\left(2c\lambda_i - \sum_{j\neq i}\frac{1}{\lambda_i - \lambda_j}\right).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\partial^2}{(\partial \lambda_i)^2} p(\lambda) &= \frac{1}{Z}\exp(-U(\lambda))\left(2c\lambda_i - \sum_{j\neq i}\frac{1}{\lambda_i - \lambda_j}\right)^2 - \frac{1}{Z}\exp(-U(\lambda))\left(2c + \sum_{j\neq i}\frac{1}{(\lambda_i - \lambda_j)^2}\right) \\
&= \left(\sum_{j\neq i}\frac{1}{\lambda_i - \lambda_j} - 2c\lambda_i\right)\frac{\partial}{\partial \lambda_i}p(\lambda) - p(\lambda)\left(2c + \sum_{j\neq i}\frac{1}{(\lambda_i - \lambda_j)^2}\right)
\end{aligned}$$

and so the right hand side of (48) is equal to

$$\alpha\sum_{i=1}^{n}\frac{\partial^2}{(\partial \lambda_i)^2}p(\lambda) = \sum_{i=1}^{n}\left(\alpha\sum_{j\neq i}\frac{1}{\lambda_i - \lambda_j} - 2\alpha c\lambda_i\right)\frac{\partial}{\partial \lambda_i}p(\lambda) - \sum_{i=1}^{n}p(\lambda)\left(2\alpha c + \alpha\sum_{j\neq i}\frac{1}{(\lambda_i - \lambda_j)^2}\right)$$

Now the left hand side of (48) is equal to

$$\begin{aligned}
&\sum_{i=1}^{n}\frac{\partial}{\partial \lambda_i}\left[\left(\alpha\sum_{i\neq j}\frac{1}{\lambda_i - \lambda_j} - \beta\lambda_i\right)p(\lambda)\right] \\
=&\sum_{i=1}^{n}\left(\alpha\sum_{i\neq j}\frac{1}{\lambda_i - \lambda_j} - \beta\lambda_i\right)\frac{\partial}{\partial \lambda_i}p(\lambda) + \sum_{i=1}^{n}\left(-\alpha\sum_{i\neq j}\frac{1}{(\lambda_i - \lambda_j)^2} - \beta\right)p(\lambda)
\end{aligned}$$

So it follows that in (48) the left-hand side is equal to the right-hand side if and only if $\beta = 2\alpha c$ or equivalently $c = \beta/2\alpha$, concluding the proof. □

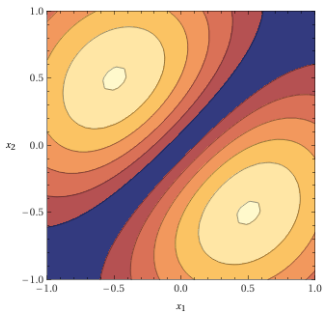


Figure 5: Plot of the invariant density of (Dyson-BM) for $d = 2$ and $\alpha = \beta = 1$

C Inferring the Eigenvector Dynamics

Theorem 3 (restated). *Denote by $(v_1(t), \dots, v_n(t))$ a tuple of orthonormal eigenvectors associated to the eigenvalues of Theorem 2. Assuming that the initial matrix $M(0)$ has simple spectrum, $v_k(t)$ satisfies for $k \in [n]$ the stochastic differential equation*

$$\mathrm{d}v_k(t) = -\frac{\alpha}{2} \sum_{\ell \neq k} \frac{1}{(\lambda_k(t) - \lambda_\ell(t))^2} v_k(t)\mathrm{d}t + \sqrt{\alpha} \sum_{\ell \neq k} \frac{1}{\lambda_k(t) - \lambda_\ell(t)} v_\ell(t)\mathrm{d}w_{\ell k}(t) \tag{Eigenvector-SDE}$$

for $\{w_{ij:i\neq j}\}$ standard Brownian motions also independent of the eigenvalue trajectories, with $w_{ji} = w_{ij}$.

Proof. Analogously to the proof of Theorem 2, we may view the v_k for $k \in [n]$ as a function of the matrix components M_{ij} for $(i,j) \in \mathcal{I} \coloneqq \{(i,j) : 1 \leq i \leq j \leq n\}$. We thus have by Itô's lemma

$$\begin{aligned} \mathrm{d}v_k &= \sum_{\eta \in \mathcal{I}} \frac{\partial v_k}{\partial M_\eta} \mathrm{d}M_\eta + \frac{1}{2} \sum_{\eta,\xi \in \mathcal{I}} \frac{\partial^2 v_k}{\partial M_\eta \partial M_\xi} \mathrm{d}M_\eta \mathrm{d}M_\xi \\ &= \sum_{\eta \in \mathcal{I}} \left(-\beta M_\eta \frac{\partial v_k}{\partial M_\eta} + \frac{1}{2} D_\eta^2 \frac{\partial^2 v_k}{(\partial M_\eta)^2} \right) dt + D_\eta \frac{\partial v_k}{\partial M_\eta} \mathrm{d}B_\eta. \end{aligned} \tag{49}$$

For the first summand of Equation (49), we observe, using Equation (37), that

$$\sum_{(i,j)\in\mathcal{I}} M_{ij} \frac{\partial v_k}{\partial M_{ij}} \mathrm{d}t = \sum_{l \neq k} \sum_{(i,j)\in\mathcal{I}} v_l^T \frac{\partial v_k}{\partial M_{ij}} v_l M_{ij} \mathrm{d}t \tag{50}$$

and further use Equation (31)

$$\begin{aligned} &= \sum_{l \neq k} \frac{1}{\lambda_k - \lambda_l} \Big(v_l^T \underbrace{\sum_{(i,j)\in\mathcal{I}} \frac{\partial M}{\partial M_{ij}} M_{ij}}_{= M \text{ by Equation (32)}} v_k \Big) v_l \mathrm{d}t \\ &= \sum_{l \neq k} \frac{\lambda_k}{\lambda_k - \lambda_l} (v_l^T v_k) v_l \mathrm{d}t. \end{aligned} \tag{51}$$

Since $l \neq k$ and the $\{v_l \mid l \in [n]\}$ are orthogonal, we get

$$= 0. \tag{52}$$

For the second summand of Equation (49), we observe

$$\frac{\partial^2 v_k}{(\partial M_\eta)^2} = \frac{\partial}{\partial M_\eta} \frac{\partial v_k}{\partial M_\eta} \tag{53}$$

and using Equation (31) and Equation (37) we get

$$= \sum_{m \neq k} \frac{\partial}{\partial M_\eta} \left[\frac{1}{\lambda_k - \lambda_m} v_m^T \frac{\partial M}{\partial M_\eta} v_k v_m \right] \tag{54}$$

by noting $\frac{\partial}{\partial M_\eta} \frac{\partial M}{\partial M_\eta} = 0$, we apply the chain rule to obtain

$$= \sum_{m \neq k} \Bigg\{ \underbrace{- \left(\frac{1}{\lambda_k - \lambda_m} \right)^2 \left(\frac{\partial \lambda_k}{\partial M_\eta} - \frac{\partial \lambda_m}{\partial M_\eta} \right) v_m^T \frac{\partial M}{\partial M_\eta} v_k v_m}_{(i)} + \underbrace{\frac{1}{\lambda_k - \lambda_m} \frac{\partial v_m^T}{\partial M_\eta} \frac{\partial M}{\partial M_\eta} v_k v_m}_{(ii)} \tag{55}$$

$$+ \underbrace{\frac{1}{\lambda_k - \lambda_m} v_m^T \frac{\partial M}{\partial M_\eta} \frac{\partial v_k}{\partial M_\eta} v_m}_{(iii)} + \underbrace{\frac{1}{\lambda_k - \lambda_m} v_m^T \frac{\partial M}{\partial M_\eta} v_k \frac{\partial v_m}{\partial M_\eta}}_{(iv)} \Bigg\}. \tag{56}$$

We now analyze each term.

Term (i) We recall Equation (33)

$$\frac{\partial \lambda_k}{\partial M_{ij}} = (v_k)_i (v_k)_j (2 - \delta_{ij}),$$

as well as that the partial derivative $\frac{\partial M}{\partial M_{ij}}$ is 0 except at i, j and j, i, where it is 1 (using symmetry).

Pulling the summation over η and all η-dependent terms in, we have

$$\begin{aligned}
&\sum_{\eta \in \mathcal{I}} (1 + \delta_\eta) \left(\frac{\partial \lambda_k}{\partial M_\eta} - \frac{\partial \lambda_m}{\partial M_\eta} \right) v_m^T \frac{\partial M}{\partial M_\eta} v_k \\
=&\sum_{ij \in \mathcal{I}} (1 + \delta_{ij}) \Bigg\{ \left[(v_k)_i (v_k)_j (2 - \delta_{ij}) - (v_m)_i (v_m)_j (2 - \delta_{ij}) \right] \\
&\qquad \left[(v_m)_i (v_k)_j + (v_m)_j (v_k)_i (1 - \delta_{ij}) \right] \Bigg\} \\
=&2 \left[v_k v_k^T v_k^T v_m - v_m^T v_m v_m^T v_k \right] \\
=&2(\delta_{km} - \delta_{km}) \\
=&0.
\end{aligned}$$

Thus, the overall contribution of term (i) is 0.

Term (ii) For term (ii) we have by Equation (31) and Equation (37)

$$(ii) = \frac{1}{\lambda_k - \lambda_m} \sum_{s \neq m} \frac{1}{\lambda_m - \lambda_s} v_m^T \frac{\partial M}{\partial M_\eta} v_s v_s^T \frac{\partial M}{\partial M_\eta} v_k v_m. \tag{57}$$

Using

$$v_m^T \frac{\partial M}{\partial M_{ij}} v_s = (v_m)_i (v_s)_j + (v_m)_j (v_s)_i (1 - \delta_{ij}), \tag{58}$$

we note that

$$\begin{aligned}
&\sum_{\eta \in \mathcal{I}} D_\eta^2 v_m^T \frac{\partial M}{\partial M_\eta} v_s v_s^T \frac{\partial M}{\partial M_\eta} v_k \\
=&\alpha \sum_{i,j \in \mathcal{I}} (1 + \delta_{ij}) \left[(v_m)_i (v_s)_j + (v_m)_j (v_s)_i (1 - \delta_{ij}) \right] \left[(v_k)_i (v_s)_j + (v_k)_j (v_s)_i (1 - \delta_{ij}) \right] \\
=&\alpha \left\{ \left[\sum_i (v_m)_i (v_k) i \right] \left[\sum_i (v_s)_i^2 \right] + v_m^T v_s v_s^T v_k \right\} \\
=&\alpha \delta_{km} + \alpha \delta_{ms} \delta_{sk} \\
=&0.
\end{aligned} \tag{59}$$

Thus, the contribution of term (ii) vanishes.

Term (iii) Analogous to term (ii).

Term (iv) By using Equation (31) and Equation (37), we get

$$\begin{aligned}(iv) =& \frac{1}{\lambda_k - \lambda_m} v_m^T \frac{\partial M}{\partial M_\eta} v_k \frac{\partial v_m}{\partial M_\eta} \\ =& \frac{1}{\lambda_k - \lambda_m} v_m^T \frac{\partial M}{\partial M_\eta} v_k \sum_{p \neq m} \frac{1}{\lambda_m - \lambda_p} v_p^T \frac{\partial M}{\partial M_\eta} v_m v_p \\ =& \frac{1}{\lambda_k - \lambda_m} \sum_{p \neq m} \frac{1}{\lambda_m - \lambda_p} \Big(v_m^T \frac{\partial M}{\partial M_\eta} v_k \Big) \Big(v_p^T \frac{\partial M}{\partial M_\eta} v_m \Big) v_p.\end{aligned}$$

By pulling in the η-dependent terms from Equation (49) which affect term (iv), and then by subsequently pulling the sum over η in, we get

$$\sum_{\eta \in \mathcal{I}} \frac{1}{2} D_\eta^2 v_m^T \frac{\partial M}{\partial M_\eta} v_k v_p^T \frac{\partial M}{\partial M_\eta} v_m \tag{60}$$

which, using Equation (59), becomes

$$\begin{aligned}=& \frac{\alpha}{2} \left(\delta_{kp} + \delta_{km} \delta_{mp} \right) \\ =& \frac{\alpha}{2} \delta_{kp}.\end{aligned}$$

Moreover, using that $p \neq m$, we can conclude that the total contribution of term (iv) is

$$\begin{aligned}& \sum_{\eta \in \mathcal{I}} \frac{1}{2} D_\eta^2 \sum_{m \neq k} (iv) \\ =& - \frac{\alpha}{2} \sum_{m \neq k} \frac{1}{(\lambda_k - \lambda_m)^2} v_k.\end{aligned}$$

Before proceeding, we prove the following lemma.

Lemma 6. *Let $v^{(k)}$ be a set of orthonormal vectors and B_{ij} a set of independent standard Brownian motions with $B_{ij} = B_{ji}$, for $i, j, k \in [n]$. We have that*

$$\tilde{B}_{lk} := (v^{(l)})^T \begin{pmatrix} \sqrt{2} B_{11} & B_{12} & B_{13} & \dots & B_{1n} \\ B_{21} & \sqrt{2} B_{22} & B_{23} & \dots & B_{2n} \\ \vdots & & \vdots & & \vdots \\ B_{n1} & B_{n2} & B_{n3} & \dots & \sqrt{2} B_{nn} \end{pmatrix} v^{(k)}$$

is a Brownian motion with variance $1 + \delta_{lk}$, that is $\tilde{B}_{lk} \overset{\text{distr}}{=} \sqrt{1 + \delta_{lk}} B$ for B a standard Brownian motion. $\tilde{B}_{lk}$ is independent of $\tilde{B}_{ab}$ for $(l, k) \neq (a, b)$ and $(l, k) \neq (b, a)$.

Proof. We have for any $a, b \in [n]$ that

$$\mathrm{d}\tilde{B}_{ab} = \sum_{i,j} (v^{(a)})_i (v^{(b)})_j \sqrt{1 + \delta_{ij}} \mathrm{d}B_{ij}.$$

Thus, $\mathbb{E}\left[\mathrm{d}\tilde{B}_{ab}\right] = 0$. We further have for $c, d \in [n]$:

$$\begin{aligned}
\mathbb{E}\left[\mathrm{d}\tilde{B}_{ab}\mathrm{d}\tilde{B}_{cd}\right] &= \sum_{ijkl}(v^{(a)})_i(v^{(b)})_j(v^{(c)})_k(v^{(d)})_l\sqrt{1+\delta_{ij}}\sqrt{1+\delta_{kl}}\,\mathbb{E}\left[\mathrm{d}B_{ij}\mathrm{d}B_{kl}\right] \\
&= \sum_{ijkl}(v^{(a)})_i(v^{(b)})_j(v^{(c)})_k(v^{(d)})_l\sqrt{1+\delta_{ij}}\sqrt{1+\delta_{kl}}\mathbb{1}_{(i,j)=(k,l)\vee(i,j)=(l,k)}\mathrm{d}t \\
&= \sum_{ij}(v^{(a)})_i(v^{(b)})_j\left\{\mathbb{1}_{i\neq j}\left[(v^{(c)})_i(v^{(d)})_j + (v^{(c)})_j(v^{(d)})_i\right] + \mathbb{1}_{i=j}(v^{(c)})_i(v^{(d)})_i\right\}(1+\delta_{ij})\mathrm{d}t \\
&= \sum_{ij}(v^{(a)})_i(v^{(b)})_j\left[(v^{(c)})_i(v^{(d)})_j + (v^{(c)})_j(v^{(d)})_i\right]\mathrm{d}t \\
&= \sum_{ij}(v^{(a)})_i(v^{(b)})_j(v^{(c)})_i(v^{(d)})_j\mathrm{d}t + \sum_{ij}(v^{(a)})_i(v^{(b)})_j(v^{(c)})_j(v^{(d)})_i\mathrm{d}t \\
&= (v^{(a)})^T v^{(c)}(v^{(b)})^T v^{(d)}\mathrm{d}t + (v^{(a)})^T v^{(d)}(v^{(b)})^T v^{(c)}\mathrm{d}t \\
&= \mathbb{1}_{(a,b)=(c,d)}\mathrm{d}t + \mathbb{1}_{(a,b)=(d,c)}\mathrm{d}t
\end{aligned} \tag{61}$$

Thus, in particular the process $\tilde{B}_{ab}$ has variance t for $a \neq b$ and variance $2t$ for $a = b$.

Since the linear combination of independent Brownian motions is jointly normal, we see from the Covariance property in Equation (61) that $\tilde{B}_{lk}$ is independent of $\tilde{B}_{ab}$ for $(l,k) \neq (a,b)$ and $(l,k) \neq (b,a)$. □

We can now turn to the third summand of Equation (49).

We have

$$\begin{aligned}
&\sum_{\eta\in\mathcal{I}} D_\eta \frac{\partial v_k}{\partial M_\eta}\mathrm{d}B_\eta \\
=&\sqrt{\alpha}\sum_{l\neq k}\frac{1}{\lambda_k-\lambda_l}v_l^T\sum_{\eta\in\mathcal{I}}\frac{\partial M}{\partial M_\eta}v_k\sqrt{1+\delta_\eta}\mathrm{d}B_\eta v_l \\
=&\sqrt{\alpha}\sum_{l\neq k}\frac{1}{\lambda_k-\lambda_l}v_l^T\begin{pmatrix}\sqrt{2}\mathrm{d}B_{11} & \mathrm{d}B_{12} & \mathrm{d}B_{13} & \dots & \mathrm{d}B_{1n}\\ \mathrm{d}B_{21} & \sqrt{2}\mathrm{d}B_{22} & \mathrm{d}B_{23} & \dots & \mathrm{d}B_{2n}\\ \vdots & & \vdots & & \vdots \\ \mathrm{d}B_{n1} & \mathrm{d}B_{n2} & \mathrm{d}B_{n3} & \dots & \sqrt{2}\mathrm{d}B_{nn}\end{pmatrix}v_k v_l \\
=&\sqrt{\alpha}\sum_{l\neq k}\frac{1}{\lambda_k-\lambda_l}v_l\mathrm{d}\tilde{B}_{lk}
\end{aligned} \tag{62}$$

where we used Lemma 6 in the last step so that $\mathrm{d}\tilde{B}_{lk}$ are symmetric Brownian motions with variance $1+\delta_{lk}$.

Having established all the terms in the SDE in Theorem 3, we check that this dynamics of the eigenvectors gives rise to normalized vectors, assuming that the initial vectors $(v_1(0), \dots, v_n(0))$ are normalized. To this end, it suffices to note that $v_k(t+\mathrm{d}t) = v_k(t) + dv_k(t)$ is normalized given that $v_k(t)$ is normalized, that is $(v_k(t))^2 = 1$ for any t. By continuity of $t \mapsto v_k(t)$, it suffices to show that $v_k(\mathrm{d}t) = v_k(0) + dv_k(0)$ remains normalized. We use the SDE from Theorem 3 to compute

that up to terms of order $\mathrm{d}t^{3/2}$ or higher we have (omitting "(0)" in the notation)

$$\begin{aligned}
&(v_k(0)+\mathrm{d}v_k(0))^2 = 1 + 2v_k^T \mathrm{d}v_k + (\mathrm{d}v_k)^2 \\
=&1 + 2\left(\frac{-\alpha}{2}\right)\sum_{l\neq k}\frac{\mathrm{d}t}{(\lambda_k-\lambda_l)^2} + 2\sqrt{\alpha}\sum_{l\neq k}\frac{1}{\lambda_l-\lambda_k}\underbrace{v_k^T v_l}_{=0}\mathrm{d}w_{lk} \\
&+\alpha\sum_{l\neq k}\sum_{j\neq k}\frac{\mathrm{d}w_{lk}\mathrm{d}w_{jk}}{(\lambda_l-\lambda_k)(\lambda_j-\lambda_k)} + \mathcal{O}(\mathrm{d}t^{3/2}) \\
=&1-\alpha\sum_{l\neq k}\frac{\mathrm{d}t}{(\lambda_k-\lambda_l)^2} + \alpha\sum_{l\neq k}\sum_{j\neq k}\frac{\delta_{jl}\mathrm{d}t}{(\lambda_k-\lambda_l)(\lambda_k-\lambda_j)} + \mathcal{O}(\mathrm{d}t^{3/2}) \\
=&1+\mathcal{O}(\mathrm{d}t^{3/2}).
\end{aligned} \tag{63}$$

Since contributions of $\mathcal{O}(\mathrm{d}t^{3/2})$ do not contribute to the dynamics for $\mathrm{d}t\to 0$, this implies that the $v_k(\mathrm{d}t)$ remain normalized. The argument may be iterated, i.e., $(v_k(2\mathrm{d}t))^2 = (v_k(\mathrm{d}t)+\mathrm{d}v_k(\mathrm{d}t))^2 = 1+\mathcal{O}(\mathrm{d}t^{3/2})$ and so on, and thus the $v_k(t)$ remain normalized for all $t>0$.

Finally, as an instructive consistency check, we also show that the eigenvectors remain orthogonal to each other (as they must, since M is symmetric for all $t\geq 0$). This can be checked from the SDE in an analogous manner using again the continuity of $t\mapsto v_k(t)$. Note that for $k\neq l$ and given $v_k^T(0)v_l(0)=0$, it suffices to show that $v_k^T(\mathrm{d}t)v_l(\mathrm{d}t)=0$. Indeed, we have (omitting "(0)" in the notation)

$$\begin{aligned}
&(v_k(0)+\mathrm{d}v_k(0))^T(v_l(0)+\mathrm{d}v_l(0)) = 0 + v_k^T dv_l + v_l^T dv_k + \mathrm{d}v_k^T dv_l \\
=&\sqrt{\alpha}\sum_{i\neq l}\frac{1}{\lambda_l-\lambda_i}v_k^T v_i \mathrm{d}w_{il} + \sqrt{\alpha}\sum_{j\neq k}\frac{1}{\lambda_k-\lambda_j}v_l^T v_j \mathrm{d}w_{jk} \\
&+\alpha\sum_{j\neq k}\sum_{i\neq l}\frac{v_j^T v_i \mathrm{d}w_{il}\mathrm{d}w_{jk}}{(\lambda_k-\lambda_j)(\lambda_l-\lambda_i)} + \mathcal{O}(\mathrm{d}t^{3/2}) \\
=&\frac{\sqrt{\alpha}}{\lambda_l-\lambda_k}\underbrace{(\mathrm{d}w_{kl}-\mathrm{d}w_{lk})}_{=0} + \alpha\sum_{j\neq k}\sum_{i\neq l}\frac{\delta_{ij}\delta_{ik}\delta_{jl}\mathrm{d}t}{(\lambda_k-\lambda_j)(\lambda_l-\lambda_i)} + \mathcal{O}(\mathrm{d}t^{3/2}) \\
=&0+\mathcal{O}(\mathrm{d}t^{3/2}).
\end{aligned} \tag{64}$$

The argument may be iterated and thus the claim holds for all $t>0$.

□

D Time reversal: Existence and Uniqueness

The time-reversal of the (Dyson-BM) in the sense of Ref. [20] is given in Eq. (3). Since the drift coefficient in the (Dyson-BM) is not locally Lipschitz continuous, the existence and uniqueness of strong solutions to (Dyson-BM) and Eq. (3), and the applicability of Ref. [20] are not obvious. While the existence and uniqueness of a strong solution to the (Dyson-BM) is well established (see, e.g., Lemma 4.3.3 in Ref. [50]), we here sketch how to ensure the other two points.

To ensure existence and uniqueness of a strong solution to Eq. (3), repeat the arguments in Lemma 4.3.3 in Ref. [50], where the divergent x^{-1}-term in the drift is replaced by the locally Lipschitz continuous approximation $\phi(x)=x^{-1}$ for $|x|\geq R^{-1}$ and $\phi(x)=xR^2$ otherwise. For any $R>0$, the desired statements follow from the local Lipschitz continuity of the drift, and details on the limit $R\to 0$ can be found in Ref. [50]. The only difference to the forward motion is the additional drift term containing the score function. However, since the dynamics is almost surely contained in the interior of the Weyl chamber [51], the propagator in the score-contribution is, as usual, dominated by white noise as $\mathrm{d}t\to 0$. Therefore, this term will not cause complications as $R\to 0$ and the arguments from Ref. [50] imply uniqueness and existence of a strong solution to Eq. (3).

Using the same regularization of the divergent term, for any $R>0$ the statements of Ref. [20] are directly applicable. Since this regularization is piecewise continuous, we can take the limit $R\to 0$ under time reversal. Since we just established the existence of the solution to Eq. (3), the limit $R\to 0$

converges. Furthermore, the arguments are expected to generalize (regularize $\phi(x) = x^{-2}$) to the eigenvector equations and their time reversal.

E Time rescaling

We rescale time for (Dyson-BM). Let $T(s)$ be a continuous, differentiable rescaling of time, monotonically increasing, with $T(0) = 0$ (for convenience).

Let $\gamma_k(t) := \lambda_k(T(t))$. With this definition, the Ito SDE Equation (Dyson-BM) corresponds to the Ito integral

$$\begin{aligned}\gamma_k(t) :=& \lambda_k(T(t)) = \lambda_k(0) + \int_0^{T(t)} \left(\alpha \sum_{i \neq k} \frac{1}{\lambda_k(s) - \lambda_i(s)} - \beta \lambda_k(s) \right) \mathrm{d}s + \int_0^{T(t)} \sqrt{2\alpha} \mathrm{d}B_k(s) \\ &= \int_0^t \left(\alpha \sum_{i \neq k} \frac{1}{\lambda_k(T(s)) - \lambda_i(T(s))} - \beta \lambda_k(T(s)) \right) T'(s) \mathrm{d}s + \int_0^t \sqrt{2\alpha T'(s)} \mathrm{d}B_k(s) \\ &\qquad \text{(using Thm. 8.5.7 in [52])} \\ &= \int_0^t \left(\alpha \sum_{i \neq k} \frac{1}{\gamma_k(s) - \gamma_i(s)} - \beta \gamma_k(s) \right) T'(s) \mathrm{d}s + \int_0^t \sqrt{2\alpha T'(s)} \mathrm{d}B_k(s) \end{aligned} \tag{65}$$

which we can write in SDE notation

$$\mathrm{d}\gamma_k(t) = \left(\alpha \sum_{i \neq k} \frac{1}{\gamma_k(t) - \gamma_i(t)} - \beta \gamma_k(t) \right) T'(t) \mathrm{d}t + \sqrt{2\alpha T'(t)} \mathrm{d}B_k(t). \tag{66}$$

By using $T(t) := \frac{1}{\alpha} t$, we obtain

$$\mathrm{d}\gamma_k(t) = \left(\sum_{i \neq k} \frac{1}{\gamma_k(t) - \gamma_i(t)} - \frac{\beta}{\alpha} \gamma_k(t) \right) \mathrm{d}t + \sqrt{2} \mathrm{d}B_k(t), \tag{67}$$

so that we can summarize the two parameters to $\eta := \frac{\beta}{\alpha}$:

$$\mathrm{d}\gamma_k(t) = \left(\sum_{i \neq k} \frac{1}{\gamma_k(t) - \gamma_i(t)} - \eta \gamma_k(t) \right) \mathrm{d}t + \sqrt{2} \mathrm{d}B_k(t) \tag{68}$$

which we call "$\gamma(\eta)$-SDE".

Dyson's conjecture says that the $\lambda(\frac{1}{N}, \frac{1}{2})$-SDE converges to global equilibrium in time $\Theta(1)$ (see [27]). Running $\lambda(\frac{1}{N}, \frac{1}{2})$ until time 1 is the same as running the $\gamma(\frac{N}{2})$-SDE until time $T(1) = \frac{1}{N}$.

F Stepsize Controller

Dyson Brownian motion almost surely never crosses the singularities. Hence, conditioning on non-crossing corresponds to conditioning on an event of probability one, which does not change the dynamics. Given the noise, we can thus calculate the maximal step size, beyond which we would cross the singularity. This is a very useful upper bound, which we employ in practice to get the numerical scheme working. It has two effects: (1) close to the boundary of the Weyl Chamber, it avoids numerically stepping over the singularities and (2) far from the boundary, it allows for larger step size, increasing efficiency. Intuitively, this mechanism thus asks "How far can we safely move before we reach a collision?"

Algorithm 2 Forward stepsize controller for Dyson SDE

1: **Input:** position $\lambda \in \mathbb{R}^n$, time $t \in \mathbb{R}^+$, independent normal samples $u \sim \mathcal{N}^n$.
2: **for** $k \in [n-1]$ **do**
3: $\quad \delta t_k \leftarrow$ maximal step size based on $\lambda_{k+1} - \lambda_k$ and samples u_k, u_{k+1} as described in Section F.1
4: **end for**
5: **Output:** stepsize $\min_i \delta t_i$.

Figure 6: Forward step size controller which exploits that non-crossing of paths happens with probability 1. The exact calculations are carried out in Section F.1.

F.1 Forward in Time

The difference between components k and $k+1$ of λ after a first-order discretization step of size $\mathrm{d}t$ reads for any time $t \in \mathbb{R}^+$

$$\Delta_k(t + \mathrm{d}t) \coloneqq \lambda_k(t + \mathrm{d}t) - \lambda_{k+1}(t + \mathrm{d}t) \tag{69}$$

$$= \lambda_k(t) - \lambda_{k+1}(t) + \mathrm{d}t f_k(t) + \sqrt{\mathrm{d}t} g_k \tag{70}$$

with functions $f_k(t) \coloneqq \alpha \left(\sum_{i \neq k} \frac{1}{\lambda_k(t) - \lambda_i(t)} - \sum_{i \neq k+1} \frac{1}{\lambda_{k+1}(t) - \lambda_i(t)} \right) - \beta \left(\lambda_k(t) - \lambda_{k+1}(t) \right)$, $g_k \coloneqq \sqrt{2\alpha} \left(X_k - X_{k+1} \right)$ where the $X_j \overset{iid}{\sim} \mathcal{N}(0,1)$ are the independent standard Gaussian random variables taken in the update step of the numerical SDE scheme. Note that Δ_k is a random variable, but if the step size is sufficiently small we must have almost surely

$$\Delta_k(t + \mathrm{d}t) > 0.$$

To find the maximal step size, we observe that the equation above is a quadratic function in the substituted $\tau \coloneqq \sqrt{\mathrm{d}t}$ yielding the inequality that

$$\tau^2 + \frac{g_k}{f_k(t)} \tau + \frac{\Delta_k(t)}{f_k(t)} \qquad \text{is} \qquad \begin{cases} > 0 & \text{if } f_k(t) > 0, \\ < 0 & \text{if } f_k(t) < 0. \end{cases} \tag{71}$$

We treat first the case $f_k(t) > 0$. The inequality is equivalent to

$$\left(\tau + \frac{g_k}{2 f_k(t)} \right)^2 > \left(\frac{g_k}{2 f_k(t)} \right)^2 - \frac{\Delta_k(t)}{f_k(t)}, \tag{72}$$

where we know that $\frac{\Delta_k(t)}{f_k(t)} > 0$ must hold. Hence, if $g_k > 0$, any stepsize $\mathrm{d}t > 0$ works. Otherwise, if $g_k < 0$, the quadratic formula shows immediately that both roots will be in the $\tau > 0$ regime. Since the parabola is convex in τ, for inequality (72) to be fulfilled, we take τ in the range from 0 to the smallest root. For $\mathrm{d}t$, that means

$$\mathrm{d}t \in \left(0, \frac{1}{4} \left(-\frac{g_k}{f_k(t)} - \sqrt{\frac{g_k^2}{f_k(t)^2} - 4 \frac{\Delta_k(t)}{f_k(t)}} \right)^2 \right). \tag{73}$$

If $f_k(t) < 0$, we have

$$\left(\tau + \frac{g_k}{2 f_k(t)} \right)^2 < \left(\frac{g_k}{2 f_k(t)} \right)^2 - \frac{\Delta_k(t)}{f_k(t)}. \tag{74}$$

Since $\frac{\Delta_k(t)}{f_k(t)} < 0$, we know that the inequality is satisfied at $\tau = 0$, and that the largest root τ_2 will be positive:

$$\tau_{1,2} = \frac{1}{2} \left(-\frac{g_k}{f_k(t)} \mp \sqrt{\frac{g_k^2}{f_k(t)^2} - 4 \frac{\Delta_k(t)}{f_k(t)}} \right).$$

Thus, any stepsize in the following interval is valid

$$\mathrm{d}t \in \left(0, \frac{1}{4}\left(-\frac{g_k}{f_k(t)} + \sqrt{\frac{g_k^2}{f_k(t)^2} - 4\frac{\Delta_k(t)}{f_k(t)}}\right)^2\right)$$

Note that $f_k(t) \neq 0$ and $g_k \neq 0$ almost surely.

F.2 Backward in Time

Backwards in time, we carry out the analogous computation for the more involved backwards SDE in Equation (3). In essence, this boils again down to solving a quadratic equation and considering all edge cases.

G Shooting mechanism

During inference, we require access to the learned score. In the forward diffusion, solely numerical errors due to the time discretisation may lead to crossings of singularities. On the one hand, when going backwards in time, due to inconsistencies in the learned score, the repulsion in Equation (3) might be too weak and the sample path may leave the Weyl Chamber for any sensible step size. Since the score is not defined outside the Weyl Chamber, this is problematic. On the other hand, the step size obtained by conditioning on not leaving the Weyl Chamber in this ill-trained point would be so small that the numerical solver would get stuck. To overcome this, we incorporate a shooting mechanism: If the repulsion force is too weak to prevent crossing of the singularity, resulting in a microscopic step size upon conditioning to remain in the Weyl Chamber, we repel with the invariant-state drift, i.e., we replace the learned score with the score in the invariant state, as we explain below. This mechanism ensures that we stay in the Weyl Chamber, while minimizing its impact on the distribution. Thus, this mechanism essentially asks "Given that we have an imperfectly learned score moving us out of the Weyl chamber, what is the safest direction to go in order to stay in the chamber?" With well-tuned parameters, the shooting gets rarely triggered (less than 0.5% of steps) but is essential, as already a single event would cause getting stuck (upon conditioning) or leaving the Weyl Chamber.

In the forward dynamics, if a certain step size would lead to the probability-0 event of leaving the Weyl-Chamber, we know that the source of the error is the finite-time-step discrete approximation, so that decreasing the step size will always provide a fix. In the backward dynamics, however, this is not guaranteed: As discussed, a possible reason for this probability-0 event in the backwards dynamics is that the score s_θ might be not perfectly learned: $s_\theta(\lambda, t) \neq s(\lambda, t)$ for some λ, t. To avoid this probability-0 event, we use the following "shooting mechanism": If we were to leave the Weyl-Chamber, we replace the learned score with the analytically known score in the invariant state, eliminating the use of the neural network at that point and leading effectively to a repulsion with the negative forward drift. This mechanism is not expected to change the dynamics in any unfavorable way since it is only applied very rarely, and only in cases where the actual learned dynamics is a much worse approximation (since it would give rise to measure zero events).

G.1 Comparison to direct sampling with eigendecompositions

One may wonder why we do not directly sample from the matrix-valued process (1) at time t, perform an eigendecomposition to obtain $\lambda(t)$, and then use the increment $\lambda(t + \mathrm{d}t)$ to learn the score via Eq. 4. While this would render learning simulation-free—relegating the Dyson SDE to the loss derivation and backward dynamics—we avoid this approach due to computational efficiency and precision. This strategy is approximately 100 times slower for graphs of size $n = 10$, as accurate eigendecomposition is computationally expensive; performing it at every step becomes prohibitively costly. Moreover, because we train on increments along the simulated trajectory, a single simulation run allows us to learn from the entire path up to time t. Our SDE implementation renders the training runtime on the order of a simulation-free diffusion model.

H Complexity dependence on graph size n of numerical step for (Dyson-BM) and (Eigenvector-SDE)

When simulating the (Dyson-BM) numerically, for a fixed eigenvalue λ_k, we need to calculate its distance to any λ_ℓ for $\ell \in [n] \setminus \{k\}$ in order to obtain the repulsion force entering the drift of the (Dyson-BM). Since we need to do this for each component, we have for a fixed time step a complexity of $\mathcal{O}\left(n^2\right)$.

When simulating (Eigenvector-SDE) numerically, for one time step, we update the entire set of n eigenvectors together. Each of the $n(n-1)/2$ entries of the Lie algebra can be computed in time $\Theta\left(1\right)$ (note that $E_{(l,k)}$ is a basis element of the Lie algebra). This update requires thus $\mathcal{O}\left(n^2\right)$. Accounting for the projection from the Lie algebra in the Lie group, this yields a complexity of $\mathcal{O}\left(n^3\right)$ for an eigenvector update.

Since (Dyson-BM) decouples from (Eigenvector-SDE), we thus have a complexity for an update step of $\mathcal{O}\left(n^2\right)$ if one is interested in only the spectrum, and a complexity of $\mathcal{O}\left(n^3\right)$ if one is interested in the spectrum and eigenvectors. Note that this modularity allows the practitioner to first learn the spectrum in isolation and do any hyperparameter tuning steps on the (smaller) spectral model, and then use this already tuned model to train and optimize for the remaining eigenvector dynamics.

In comparison, an OU-based approach on the entire graph with a GNN-based algorithm requires for a fixed time point an update of the SDE in n^2 entries in time $\mathcal{O}\left(n^2\right)$ followed by C message passing steps (a GNN is trained using message passing, where C is a constant defining how many hops a message is being passed), with each message passing taking time $\mathcal{O}\left(n^3\right)$, leading to an overall complexity of $\mathcal{O}\left(n^3\right)$ per time step, assuming that C is constant.

I Empirical Considerations

While we give a thorough analysis of the computational complexity of each update step, the number of time steps is determined by the adaptive step size algorithm. For practical reasons, our method can be configured with a minimal (and maximal) step size. We illustrate the distribution of the number of time steps on the Brain dataset in Figure 7, where it can be observed that the number of time steps taken concentrates very well. Note that this number of time steps perfectly suffices even for a dataset as large as Brain, which contains $15,000$ graphs. Note further that while these are the number of steps the numerical integrator takes forward, learning has to happen only on much fewer steps, through the described pre-defined schedule. In the case of Brain, learning happens only on 651 time points. The numerical generation of $105,000$ paths with step number depicted in Figure 7 took in total 5 seconds. For memory considerations, note that we require for training only a single A100 GPU (80 GB Video Random Access Memory).

We further note that the skip event (see Algorithm 1) is indeed a rare event: In the Brain setting, the empirical probability of a step being a skip is 10^{-5}. Similarly, shootings in the analogous backwards runs happen rarely. Note that skips and shoots are needed for correctness and stability, but no negative impact on runtime due to their rarity.

J Inference

We describe in Algorithm 3 the sampling procedure with the shooting mechanism (see Section G for details) incorporated, ensuring that we remain in the Weyl Chamber.

K Engineering

We follow [53] by using SiLU activations. We further use EMA to average over multiple runs [54].

A key strength of DyDM is that the score $s_\theta(\lambda, t)$ can be parameterized with any learning architecture, without being constrained to GNNs or graph transformers. We demonstrate this by parameterizing the score with a simple MLP. The size of layers varies by application, and we document it for each dataset in the uploaded Supplementary Material. For instance, for the large dataset of $15,000$ brain ego graphs, our model consists of a hidden MLP of depth 4, where the input and output layer have width

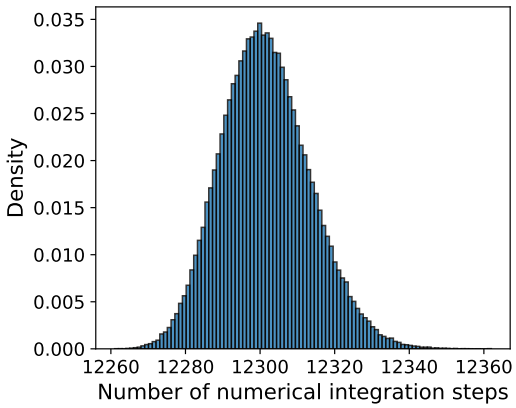


Figure 7: Concentration of time steps of the numerical integrator: Within 5 second, we generated $105,000$ forward paths using our numerical scheme, where the total number of steps per path are distributed as depicted in this histogram. As can be seen, the distribution concentrates well.

Algorithm 3 DyDM sampling

Input: dimension n, schedule $\mathcal{T} = \{t_j\}$, trained score network s_θ
$\lambda(T) \leftarrow$ Spectrum of a random GOE matrix. ▷ Sampling from invariant distribution.
$t \leftarrow T$
while $t > 0$ **do**
  Let $u \sim \mathcal{N}(0, I_n)$
  $\delta t \leftarrow$ BackwardStepsizeController$(\lambda(t), u)$ ▷ Step size as described in F.2.
  $\hat{s}_\theta \leftarrow$ Interpolation of $s_\theta(\lambda, t_j)$ and $s_\theta(\lambda, t_{j+1})$ for $t_j, t_{j+1} \in \mathcal{T}$ closest points in schedule
  $\mathrm{BackwardsDrift} \leftarrow$ drift of Equation (3) at time t and point $\lambda(t)$ with score $\hat{s}_\theta(\lambda, t_{j+1})$
  if $\lambda_t - \mathrm{BackwardsDrift} \cdot \delta t - u\sqrt{2\alpha\delta t} \in$ Weyl Chamber **then**
    $\lambda_{t-\delta t} \leftarrow \lambda_t - \mathrm{BackwardsDrift} \cdot \delta t - u\sqrt{2\alpha\delta t}$
  else ▷ Otherwise – in a rare event – shooting mechanism triggers (see Section G)
    $\lambda_{t-\delta t} \leftarrow \lambda_t - (-\mathrm{ForwardDrift}) \cdot \delta t - u\sqrt{2\alpha\delta t}$ ▷ ForwardDrift as in 3.1
  end if
  $t \leftarrow t - \delta t$
end while
Output: spectral sample $\lambda(0) \in \mathbb{R}^n$

Figure 8: Sampling from the Dyson Diffusion Model. The Equation (3) is evolved backward in time with the shooting mechanism and an adaptive step size, ensuring that the paths remain in the Weyl Chamber.

64 and the hidden layers have width 256. The space + time data is first scaled up with a linear layer from size $n + 1 = 11$ followed by a batch norm to feed into the hidden MLP, and upon processing through the hidden MLP, it gets scaled down through a simple linear layer to size $n = 10$. In the hidden MLP, we employ as nonlinearities scaled SiLU functions, as argued by [53]. We use AdamW for optimization [55].

To make full use of the GPU memory, in one epoch, we sample N_{paths} paths in parallel. If the dataset is too small to fill the GPU memory, we sample multiple, independent, paths of the same data points in parallel. From these samples, we update the score network s_θ in smaller batches. Proceeding in this way ensures that JAX uses the full potential of the GPU. These parameters (N_{paths} and batch size) can be specified in the configuration file.

Our runtimes are as follows: For WL-Bimodal, sampling 1000 paths takes 16 seconds and training takes 19 hours. For Community Small, sampling 1000 paths takes 60 seconds and training takes

10 hours. For Brain, sampling of 1000 paths is very fast at 11 seconds. Training takes 16 hours. Runtime-wise, our strength therefore lies in fast generation.

We sample 1000 paths to make the reported distances statistically robust, and report in addition to the mean also the Wasserstein of the marginals of these 1000 samples to the ground truth, accounting both for high-dimensional and marginal effects (see Section 4.2).

We implement the model in JAX [56] (version 0.9.0, licensed under Apache license, version 2.0) and Equinox `https://github.com/patrick-kidger/equinox` [57] (version 0.13.6, licensed under Apache license, version 2.0).

K.1 Choice of Time Grid

As outlined in Section 3.2, we choose an exponential time grid on which the objective is learned. This is due to the mixing behavior of Dyson's Brownian motion. For instance, for the Brain dataset, we use an exponential time grid as detailed in Table 3.

Table 3: Example of exponential time grid, here for the brain dataset which contains in total $15,000$ graphs. $\mathrm{d}t$ is 0.05, and the final time is $T = 12.0$.

from	**to**	**stepsize**
0	$1/8$	$1/64 \cdot \mathrm{d}t$
$1/8$	$1/4$	$1/32 \cdot \mathrm{d}t$
$1/4$	$1/2$	$1/16 \cdot \mathrm{d}t$
$1/2$	1	$1/8 \cdot \mathrm{d}t$
1	2	$1/4 \cdot \mathrm{d}t$
2	3	$1/2 \cdot \mathrm{d}t$
3	7	$1 \cdot \mathrm{d}t$
7	T	$2 \cdot \mathrm{d}t$

K.2 Preprocessing and dealing with different dimensions

As is the case for an Ornstein-Uhlenbeck diffusion, the speed of convergence depends for (Dyson-BM) on (1) the coefficients and (2) the initial condition. We can thus choose to significantly vary the final time T or rescale the initial condition. We choose the latter (although both options are feasible). To that end, we rescale the spectra with an affine transformation in a preprocessing step. In more detail: For a given set of graphs, we perform an eigendecomposition, and rescale so that among the entire dataset the largest eigenvalue is at most $\lambda_{\max}$ and at least $\lambda_{\min}$. For instance, on the benchmark models we chose $\lambda_{\max} = 5$, $\lambda_{\min} = -5$. If a spectrum has eigenvalues of multiplicities greater than 1, we perform an ϵ-perturbation, where the ϵ depends on the distance of the closest eigenvalues in the dataset. In postprocessing, the spectra are scaled back and the perturbation is undone for eigenvalues that are at most ϵ apart after generation. Note that with this preprocessing, only one eigendecomposition per training graph is necessary.

L Challenges of Dyson's Brownian Motion

Using Dyson's Brownian motion for a diffusion presents several challenges, all of which we overcame in this paper. First, the Dyson SDE is not an OU process, but instead an SDE with singularities of order $\mathcal{O}(1/(\lambda_k - \lambda_l))$ in the drift, posing both theoretical and numerical challenges. Second, the conditional density $p(x \mid x_0)$ is non-Gaussian and challenging to obtain, as with any non-OU diffusion process. Therefore, we do not have access to a canonical loss function. Finally, the non-availability of conditional distributions means that training is not simulation-free.

We overcome the obstacles mentioned above and provide a diffusion model for the spectra of graphs based on the Dyson SDE (Fig. 1). The model is not only efficient but is also able to distinguish between spectra of graphs that GNNs are blind to (Fig. 2). In addition, with DyDM, no ad hoc data augmentation is necessary. Further, through (Eigenvector-SDE), we give the dynamics of the remaining information in form of conditioned eigenvector dynamics, hence making them accessible for future work devoted to eigenvector diffusion.

L.1 Why not log-transform the SDE?

One idea could be to transform the spectral SDE into terms $\lambda_1, \lambda_2 - \lambda_3, \ldots, \lambda_{n-1} - \lambda_n$ and take logarithms, to avoid singularities. However, this is not desirable for multiple reasons: First, upon applying Ito to this SDE, we (i) lose the $\ln$ and obtain singularities again and (ii) get higher order singularities $\mathrm{d}\ln(x_t) = \frac{1}{x_t}\mathrm{d}x_t - \frac{1}{2x_t^2}(\mathrm{d}x_t)^2$. Second, the space that would need to be sampled would certainly not decrease, since now the transformed domain reaches from $-\infty$ to $+\infty$. Hence, we choose the method described in the main part.

M Why not learn on all $n!$ many graph representations?

In short, learning $n!$ more data is much harder. This point has been mentioned by previous literature, e.g. [8]. However, if we go deeper, the interested reader might wonder why exactly.

M.1 Rigorous argument

We give here a toy example, where the challenge can be phrased rigorously. Suppose we have a binary matrix $X \in \{0,1\}^{m,k}$ with independent entries, which are for $i \in [m], j \in [k]$ distributed as $X_{ij} \sim \mathrm{Bernoulli}\,(p_j)$ for some unknown $p_j \in [0,1]$. The task is to estimate the p_j. The motivation for this example stems from the following setting: We want to learn the probability of k objects (for instance, graphs), each having m representations (for instance, representations of graphs such as adjacency matrices). Each column of the matrix X thus consists of all representations of the same object. We define two estimators, with est_A using the inductive bias and est_B not using it.

To that end, suppose we have N uniformly at random obtained samples $Z_1, \ldots, Z_N$. In more detail, that means that we sample for each $\ell \in [N]$ a pair of indices $(i_\ell, j_\ell) \in [m] \times [k]$ uniformly at random, and describe the obtained sample by $Z_\ell \sim X_{i_\ell, j_\ell}$. Estimator est_A makes use of the inductive bias. That is, for $u \in [m], v \in [k]$ we define

$$\mathrm{est}_\mathrm{A}^{(\mathrm{u},\mathrm{v})} \coloneqq \frac{k}{N} \sum_{\ell \in [N]} Z_\ell \mathbb{1}_{j_\ell = v}.$$

Estimator est_B does not make use of the inductive bias. That is, for $u \in [m], v \in [k]$, we define

$$\mathrm{est}_\mathrm{B}^{(\mathrm{u},\mathrm{v})} \coloneqq \frac{k \cdot m}{N} \sum_{\ell \in [N]} Z_\ell \mathbb{1}_{i_\ell = u, j_\ell = v}.$$

Clearly, both estimators are unbiased: $\mathbb{E}\left[\mathrm{est}_\mathrm{A}^{(\mathrm{u},\mathrm{v})}\right] = \mathbb{E}\left[\mathrm{est}_\mathrm{B}^{(\mathrm{u},\mathrm{v})}\right] = p_v$. However, their mean squared error, defined for $X \in \{A, B\}$ as

$$\mathrm{MSE}(\mathrm{est}_\mathrm{X}) \coloneqq \frac{1}{m \cdot k} \sum_{u \in [m], v \in [k]} \mathbb{E}\left[\left(\mathrm{est}_\mathrm{X}^{(\mathrm{u},\mathrm{v})} - p_v\right)^2\right],$$

varies significantly between est_A and est_B.

Theorem 7 (Mean square error)**.** *In dependence of the problem size m and number of samples N, the mean square error of est_A is of order $\Theta(1/N)$, while the mean square error of est_B is of order $\Theta(m/N)$.*

To prove Theorem 7, we derive the mean squared error for both estimators.

Lemma 8 (MSE of est_A)**.** *The mean squared error of estimator est_A is*

$$\mathrm{MSE}\left(\mathrm{est}_\mathrm{A}\right) = \frac{1}{N} \sum_{v \in [k]} \left(p_v - \frac{1}{k} p_v^2\right).$$

Proof. We have for $u \in [m], v \in [k]$

$$\begin{aligned}
&\mathbb{E}\left[\left(\mathrm{est}_{\mathrm{A}}^{(\mathrm{u},\mathrm{v})} - p_v\right)^2\right] \\
=&\mathbb{E}\left[\left(\frac{k}{N}\sum_{\ell\in[N]} Z_\ell \mathbb{1}_{j_\ell=v} - p_v\right)^2\right] \\
=&\frac{k^2}{N^2}\sum_{\ell\in[N]}\mathbb{E}\left[\sum_{o\in[N]} Z_\ell \mathbb{1}_{j_\ell=v} X_o \mathbb{1}_{j_o=v}\right] - 2\frac{k}{N}\,\mathbb{E}\left[\sum_{\ell\in[N]} Z_\ell \mathbb{1}_{j_\ell=v}\right] p_v + p_v^2 \\
=&\left(1-\frac{1}{N}\right)p_v^2 + \frac{k}{N}p_v - 2p_v^2 + p_v^2 \\
=&\frac{k}{N}p_v - \frac{1}{N}p_v^2.
\end{aligned}$$

Averaging over all $u \in [m], v \in [k]$, we obtain the desired result. □

Lemma 9 (MSE of $\mathrm{est}_{\mathrm{B}}$)**.** *The mean squared error of estimators* $\mathrm{est}_{\mathrm{B}}^{(\mathrm{u},\mathrm{v})}$ *for* $u \in [m]$*,* $v \in [k]$ *is*

$$\mathrm{MSE}\left(\mathrm{est}_{\mathrm{B}}\right) = \frac{1}{N}\sum_{v\in[k]}\left(m \cdot p_v - \frac{1}{k}p_v^2\right).$$

Proof. We have for $u \in [m], v \in [k]$

$$\begin{aligned}
&\mathbb{E}\left[\left(\mathrm{est}_{\mathrm{B}}^{(\mathrm{u},\mathrm{v})} - p_v\right)^2\right] \\
=&\mathbb{E}\left[\left(\frac{k\cdot m}{N}\sum_{\ell\in[N]} Z_\ell \mathbb{1}_{i_\ell=u, j_\ell=v} - p_v\right)^2\right] \\
=&\frac{k^2m^2}{N^2}\sum_{\ell\in[N]}\mathbb{E}\left[\sum_{o\in[N]} Z_\ell \mathbb{1}_{i_\ell=u,j_\ell=v} Z_o \mathbb{1}_{i_o=u,j_o=v}\right] - 2\frac{k\,m}{N}\,\mathbb{E}\left[\sum_{\ell\in[N]} Z_\ell \mathbb{1}_{i_\ell=u,j_\ell=v}\right] p_v + p_v^2 \\
=&\left(1-\frac{1}{N}\right)p_v^2 + \frac{k\,m}{N}p_v - 2p_v^2 + p_v^2 \\
=&\frac{k\,m}{N}p_v - \frac{1}{N}p_v^2
\end{aligned}$$

Summing over all estimators for $u \in [m], v \in [k]$ gives the desired result. □

By considering the estimation problem as a problem of parameters m, N, Theorem 7 follows directly from Lemma 8, Lemma 9.

The same argument may be carried out with Normal instead of Bernoulli random variables.

N WL-equivalence of regular graphs

We now prove Lemma 1.

Lemma 1 (restated). *For fixed* $n, k \in \mathbb{N}$*, all* k*-regular graphs* $G \in \mathcal{G}^n$ *are WL equivalent. Every graph* $G \in \mathcal{G}^n$ *that is WL equivalent to a* k*-regular graph is* k*-regular.*

Proof. Recall that the 1-Weisfeiler-Leman (WL) algorithm tests graph equivalence by iteratively updating vertex *colors*. Initially, all vertices share one color. In each step, a vertex's new color is determined by the multiset (a set allowing for duplicates) of its own color and its neighbors' current

colors. This continues until the coloring stabilizes. Two graphs are WL-equivalent if this process generates identical color counts (histograms) at every step.

For a k-regular graph, since every vertex has exactly k neighbors, in each iteration of the WL-algorithm all vertices have the same color. Thus in particular the histograms are always the same and therefore any two k-regular graphs are WL-equivalent.

A k-regular graph is not WL-equivalent to a ℓ-regular graph for $k \neq \ell$ since the colors after the first iteration are distinct as the number of neighbors is different. Moreover, if a graph is not regular, then the first iteration must assign a different new color to at least two vertices in the first iteration. Therefore, the color histogram is not the same as the color histogram of a regular graph and thus they are not WL-equivalent. This shows that the WL-equivalence class of a k-regular graph is exactly the set of k-regular graphs. □

In particular, it follows from Morris [14] that GNNs cannot distinguish k-regular graphs.

O Laplacian Spectrum

The Laplacian spectrum captures key properties of a graph, such as the Fiedler Value encoding the algebraic connectivity of the graph. Our model works perfectly well on the Laplacian spectrum, for which we plot the marginal densities on the datasets "Brain" and "Community Small". Note that the model perfectly learns key properties such as the fact that the smallest eigenvalue is 0 and learns the distributions well.

For Brain (Figure 9), we plot for each eigenvalue λ_i a histogram of the DyDM-generated marginal (blue) compared to a histogram of the Ground Truth (note that the Brain dataset test for generalization). For the commonly used "Community Small" dataset, we show in Figure 10 the first 12 eigenvalues, since only 73 graphs have more than 12 nodes.

Note that in this Section, we use the convention $\lambda_1 \leq \cdots \leq \lambda_n$ to stay consistent with Graph Laplacian literature.

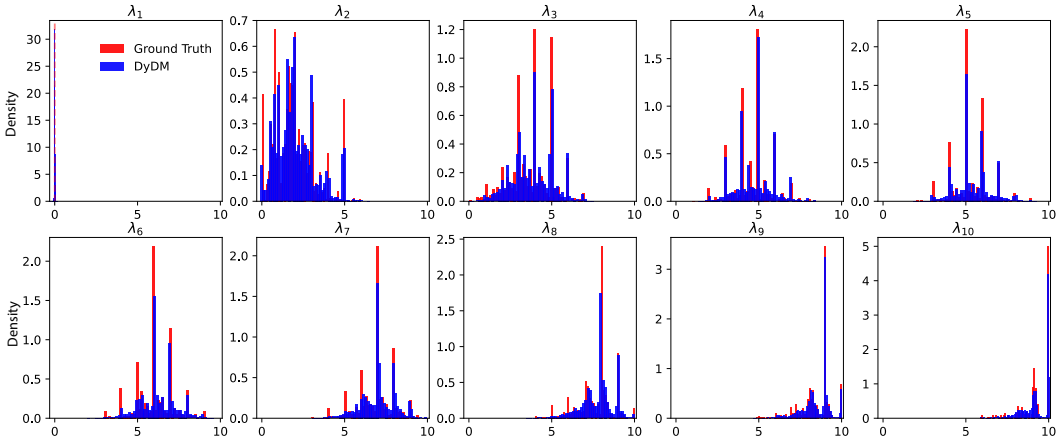


Figure 9: Performance of DyDM on the Laplacian of the Brain dataset compared to the underlying distribution: DyDM perfectly learns the (rough) distribution and key properties such as the smallest eigenvalue always being 0 (faint line, since it is very concentrated) or the distribution of the Fiedler Value (λ_2).

P Datasets

WL-Bimodal The WL-Bimodal dataset consists of 80% graph A and 20% graph B (see Fig. 2) adjacency matrices. We drew among all permutations $5,000$ permutations uniformly at random and shuffled the graphs. Hence, we have $5,000$ matrices, of which $4,000$ represent graph A and $1,000$

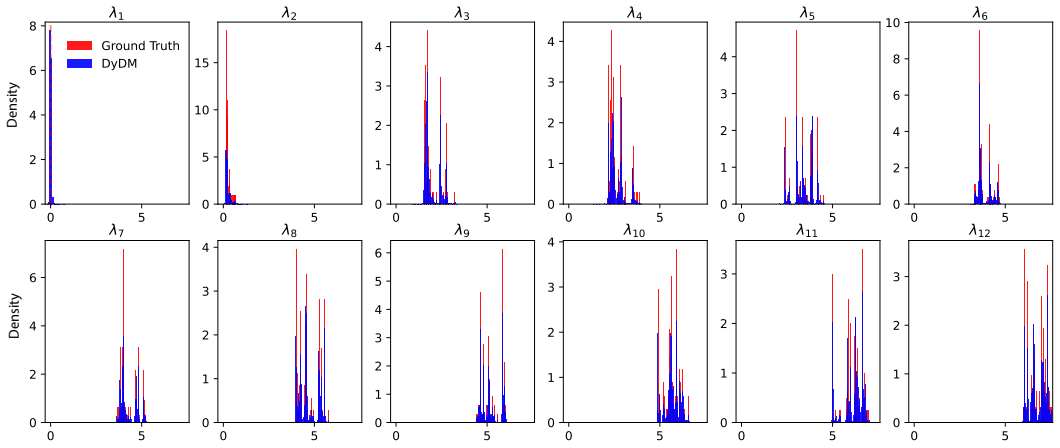


Figure 10: Performance of DyDM on the Laplacian of the Community Small dataset: DyDM learns the distribution and key properties very well.

represent graph B. Adjacency matrices representing the same graph are not identical matrices, but rather permutations of each other, making this dataset relevant to *graph* generative models. The first 80% of this dataset are used for training, the remaining 20% are used for testing.

Community-small This standard benchmark dataset [8, 9, 33] consists of 100 graphs of size up to 20 vertices. We comment in Section Q.1 on the small dataset (100 graphs) compared to the big dimension (up to 20 vertices) and the thereby induced effect of undersampling.

Brain We report this dataset in our repository. In detail, we construct from the brain graph [34, 35] so-called ego-graphs. That is, we take the (distance 1) neighborhoods of vertices, and consider the induced subgraph. From those, we generate $15,000$ graphs of size $n = 5$ to $n = 10$ vertices with eigenvalue multiplicity up to 3, with the closest eigenvalues – which are not multiplicities – having distance 0.036. We take 70% as train graphs, 15% as validation, and the remaining 15% as test graphs.

Q Comparing to Benchmark Models

Q.1 On undersampling

For the "bimodal" case, we have sufficient statistics for the 10-dimensional space C_{10} ($N = 5,000$ graph samples, each isomorphic to one of two graphs) and know in addition the underlying distribution; hence, an extensive interpretation of this result is appropriate. For a fair comparison, we thus follow the standard test/train split procedure as reported in [9, 33, 8] using 80% of the data as train data and the remaining 20% as test data.

Conversely, the standard benchmark set "community small" [8, 9, 33] contains *only* 100 graphs, and each has a size of up to $n = 20$ vertices. Thus, a comparison from the learned distribution based on (few) training samples to (very few) test samples suffers from undersampling. This becomes very stark if one considers the following issue: If one would extract 80% of the dataset for training, *where* they are taken from already matters a lot: Whether they are taken from the front or back changes the maximal graph size in the training set. Depending on whether the training data is taken from the front or back of the dataset, a model trained on the training set might thus have no possibility to learn the correct maximal graph size. More generally, poorness of benchmarks in graph generative learning has been recently addressed by [32]. To offer some consistent comparison, we do include the standard benchmark "community small", but focus on *memorization* rather than the (on those benchmarks untestable) generalization.

To overcome the issue of undersampling, we construct a set of $15,000$ ego-graphs from the brain dataset [34] as described in Section P, which is sufficiently large to *not* suffer from undersampling. We train both our model and DiGress on 70% (= $10,500$ graphs), perform hyperparameter tuning (see below for details on the DiGress hyperparameter tuning) on a validation set of 15%, and test on the remaining 15%.

Q.2 GDSS

To compare to the GDSS model [9], we take the following approach to ensure a fair comparison: The GDSS model has been trained and optimised on the ego small and community small dataset, so that we take for these datasets the snapshots and hyperparameters given by the original paper [9]. For the remaining datasets, we start with the settings from the community-small dataset since that has similar size, adapt the maximal number of vertices to the dataset, and then perform hyperparameter tuning as described in Appendix C of the GDSS paper: We form a grid search on the model's following parameters: The scale coefficient in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, the signal-to-noise ratio in $\{0.05, 0.1, 0.15, 0.2\}$, and in addition to the GDSS paper, we also try different β_{max} in $\{1, 10, 20\}$, and try different batch sizes $\{128, 4096\}$. We test with and without EMA. The motivation for the additional parameters we hyper tune is that we observed that they further improve the GDSS model. This gives full fairness to the model. The training time on one A100 GPU per job was for the brain dataset about 24 hours for each of the configurations with batch size 128 and approximately 1 hour and 45 minutes for each of the configurations with batch size 4096.

For the community small dataset, training took 15 minutes, sampling 450 seconds.

For the WL-Bimodal case, again using an $A100$ GPU, one training run of GDSS took 5 hours, and one sampling run of $1{,}000$ graphs took 450 seconds.

On the Two-WL graph case, our hyperparameter tuning resulted in 280 combinations that we trained and sampled from. From each of those 280 models, we generated $1,000$ graphs. Most of those models worked fine, that is, 269 models did not contain NaNs in their output. From those, we select the best model based on the following relative error: From the generated samples, we calculate the share of spectra ϵ-close to the spectrum of graph A, say $\hat{p}_A$ and the share of spectra ϵ-close to graph $\hat{p}_B$ (we choose the l_2 distance with $\epsilon = 0.2$). Recall that in the ground truth, we have $p_A = 0.8$, $p_B = 0.2$. We then selected the best model based on the relative error

$$\frac{|\hat{p}_A - p_A|}{p_A} + \frac{|\hat{p}_B - p_B|}{p_B}.$$

We then trained the best hyperparameter choice with 5 different seed values and report their mean and standard deviation.

In summary, we invested a lot of resources in following both the hyperparameter tuning given in the GDSS paper and, in addition, tried new hyperparameters, leading to the 280 models that we trained and sampled from. This ensures maximal fairness.

Q.3 EDP-GNN

We proceeded analogously to GDSS: We used the given configurations for community small. We observe that the model already performs hyperparameter tuning of the noise scales during sampling. For the other datasets, we have used the given configurations for community small and in addition tried the learning rates $\{0.001, 0.0002\}$, number of diffusion steps $\{1,000, 2,000\}$ and number of layers $\{4, 6\}$. In the two graph case, for example, the optimal configuration was with learning rate 0.0002, number of diffusion steps being equal to $2,000$ and 6 layers. The training time for $5,000$ epochs on a H100 GPU was approximately 10 hours.

Q.4 DiGress

As with the previous models, we used the given configurations for community small. For the other datasets, we have used the given configurations for community small and in addition tried the learning rates $\{0.001, 0.0002\}$, weight decay parameters $\{10^{-2}, 10^{-12}\}$, number of diffusion steps $\{500, 1,000\}$ and number of layers $\{5, 8\}$. The model was quite robust to hyperparameter tuning and in the two graph example, for all parameters the model sampled the 80%-graph with likelihood

between 75% and 82%. The training time for $1,000$ epochs on a H100 GPU was approximately 3 hours.

Q.5 ConGress

Since ConGress and DiGress come from the same paper, we have proceeded almost exactly as in DiGress. We tried the learning rates $\{0.001, 0.0002\}$, weight decay parameters $\{10^{-2}, 10^{-12}\}$, number of diffusion steps $\{500, 1,000\}$ and number of layers $\{6, 8\}$. The model was again rather robust to hyperparameter tuning, yet not as much as DiGress. In the two graph example, for all parameters the model sampled the 80%-graph with likelihood between 10% and 25%. The default learning rate from the community-small configuration was 0.0002, yet we have observed that the results were significantly better with learning rate 0.001. The training time for $1,000$ epochs on a H100 GPU was approximately 3 hours.

Q.6 Comparison table with more digits

We provide here the table shown in the main part Table 1 but with more digits (*not* implying that all are statistically significant): This rationalizes which entries in Table 1 are dark green and which ones are light green. Note that this result is included only for transparency, since the reported digits here go beyond the significant digits.

Table 4: Statistical distances of DyDM compared to standard models, as in Table 1 but with more digits.

Dataset	WL-Bimodal		Community Small		Brain	
Distance	μ	$\mathcal{W}_{\text{marg}}$	μ	$\mathcal{W}_{\text{marg}}$	μ	$\mathcal{W}_{\text{marg}}$
DyDM (ours)	**0.0166**	**0.0076**	**0.0671**	**0.0172**	**0.0455**	**0.0275**
EDP-GNN	0.1342	0.0750	0.4164	0.1356	0.0723	0.0321
GDSS	0.2416	0.1339	0.3812	0.1444	0.3276	0.1191
ConGress	0.3802	0.1590	0.2741	0.1138	0.1315	0.0296
DiGress (no trick)	1.0568	0.2852	2.5088	0.4481	0.5686	0.1749
DiGress (trick)	0.0302	**0.0073**	0.0934	0.0254	0.1208	**0.0285**

Table 5: Empirical standard deviation of runs.

Dataset	WL-Bimodal		Community Small		Brain	
Distance	μ	$\mathcal{W}_{\text{marg}}$	μ	$\mathcal{W}_{\text{marg}}$	μ	$\mathcal{W}_{\text{marg}}$
DyDM (ours)	0.0057	0.0011	0.0341	0.0044	-	-
EDP-GNN	0.0251	0.0226	-	-	0.0838	0.0265
GDSS	0.0125	0.0067	0.0617	0.0055	0.0951	0.0099
ConGress	0.3143	0.0736	0.0171	0.0033	0.0943	0.0131
DiGress (no trick)	0.0349	0.0060	0.0169	0.0057	0.4252	0.1293
DiGress (trick)	0.0150	0.0033	0.0129	0.0025	0.0458	0.0065

R Learning dynamics of EDP-GNN

We report in Figure 11 the learning progress of EDP-GNN on the WL-Bimodal dataset. We average over 4 different training and sampling runs of EDP-GNN. After $5,000$ epochs, we observe the result of EDP-GNN reported in Figure 2.

We observe that the model quickly learns the WL equivalence class (the pink line is from epoch 500 onward close to 1). The share of graph A and graph B samples initially increase until epoch $1,500$, but then remain low and significantly different from the ground truth (blue and green dashed lines). Importantly, a significant share of the samples are WL-equivalent but neither isomorphic to graph A nor to graph B.

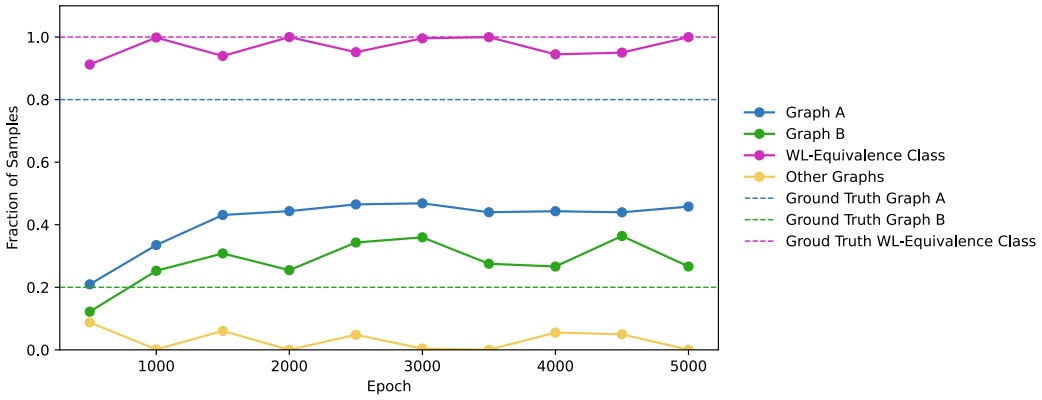


Figure 11: Learning dynamics of EDP-GNN on the WL-Bimodal dataset (ground truth = 80% graph A, 20% graph B): The model learns very quickly (after less than 500 epochs) the WL-equivalence class, but struggles to learn graphs A and B.

S Diffusion Model for the Eigenvector SDE

In this appendix, we discuss a diffusion model for the Eigenvector SDE. First, in section S.1 we present some preliminaries on SDEs on $\mathrm{O}(n)$. In section S.2 we show that the Stratonovich SDE in (5) agrees with the (Eigenvector-SDE). In section S.3 we establish the time-reversal formula for the eigenvector SDE, and in section S.4 we explain the corresponding loss function.

Foundational results on diffusion models on Riemannian manifolds were established by [26], treating the case of Brownian motions with a drift. Moreover, recently [28] discussed diffusion models for Lie groups acting on $\mathbb{R}^n$. However, a diffusion model for SDEs of the type of the (Eigenvector-SDE) has not been studied in the literature so far, and in this appendix we establish diffusion model for more general SDEs on Lie groups.

We denote by $\mathrm{O}(n)$ the orthogonal group

$$\mathrm{O}(n) = \{X \in \mathrm{GL}_n(\mathbb{R}) \,:\, X^T = X^{-1}\},$$

where $\mathrm{GL}_n(\mathbb{R}) = \{X \in \mathbb{R}^{n\times n} \,:\, \det(X) \neq 0\}$ is the group of invertible $n \times n$ matrices. The Lie algebra of $\mathrm{O}(n)$ is the tangent space at the identity and can be calculated to be

$$\mathfrak{o}(n) = T_e\mathrm{O}(n) = \{Z \in \mathbb{R}^{n\times n} \,:\, Z^T = -Z\}.$$

Moreover, it is then easily seen that the tangent space at $X \in \mathrm{O}(n)$ is given by

$$T_X\mathrm{O}(n) = X\mathfrak{o}(n) = \{XZ \,:\, Z \in \mathfrak{o}(n)\}.$$

The key insight to implement an efficient diffusion model for (Eigenvector-SDE) is to rewrite the SDE as an exponential update rule that makes use of the Lie algebra. To that end, we first rewrite the SDE as a Stratonovich SDE on the Lie group $\mathrm{O}(n)$. Denote by $\mathfrak{o}(n) = T_{\mathrm{Id}}\mathrm{O}(n) = \{A \in \mathbb{R}^{n\times n} \,:\, A^T = -A\}$ the Lie algebra of $\mathrm{O}(n)$, which is the tangent space of $\mathrm{O}(n)$ at the identity matrix. Let $E_{(\ell,k)} \in \mathfrak{o}(n)$ for $1 \leq \ell < k \leq d$ be the matrix that is 1 at the (ℓ,k)-entry, -1 at (k,ℓ), and 0 otherwise. We prove in Proposition 4 that the (Eigenvector-SDE) $X(t) = (v_1(t), \ldots, v_n(t)) \in \mathrm{O}(n)$ yields on $\mathrm{O}(n)$ the Stratonovich SDE (see also (5)

$$dX(t) = \sqrt{\alpha} \sum_{1\leq \ell<k\leq n} \left(\frac{X(t)E_{(\ell,k)}}{\lambda_k(t) - \lambda_\ell(t)}\right) \circ \mathrm{d}W^{(\ell,k)}(t), \tag{75}$$

which allows to derive the numerical update step for small $h > 0$, $X(t+h) \approx X(t)\exp(Z)$ [29], where $Z = \sqrt{\alpha h}\sum_{\ell<k} \frac{E_{(\ell,k)}\mathcal{N}^{(\ell,k)}(0,1)}{\lambda_k(t)-\lambda_\ell(t)} \in \mathfrak{o}(n)$, $\mathcal{N}^{(\ell,k)}(0,1)$ are independent samples of standard 1-dimensional Gaussians, and $\exp(Z) = \sum_{i=0}^{\infty} \frac{A^i}{i!}$ is the matrix exponential. For $h = \mathrm{d}t$, the normal

samples $\sqrt{h}\mathcal{N}^{(\ell,k)}(0,1)$ become the Wiener increments $\mathrm{d}W_{\ell,k}$ in Sec. 3.3. The Orthogonal group $O(n)$ is not connected and splits into the sets $SO(n)$ with determinant $+1$ and $O(n) \setminus SO(n)$ with determinant -1. The update equation $X(t+h) \approx X(t)\exp(Z)$ retains the sign of the determinant $\det(X(t+h)) = \exp(\mathrm{tr}(Z))\det(X(t)) = \det(X(t))$. Since any data point $\det(X(0)) \in \{\pm 1\}$, the initial sign determines which connected component the dynamics remain in. Consequently, the share of determinant ± 1 must be reflected in the stationary distribution. Note that Z does not explicitly depend on X, and hence the dynamics is rotation invariant, i.e. $R(X(t+h)) = (RX)(t+h)$ where the latter corresponds to propagation in the rotated space. Therefore, the stationary state modulo sign of the determinant must be the uniform distribution on $SO(n)$. Alternatively, one could permute rows in the input data $X(0)$ to fix positive sign and thereby evolution on $SO(n)$.

While SDEs of the form (75) on $\mathrm{O}(n)$ have not been studied for diffusion models so far, we observe that the latter SDE is, as explained in App. S.3, similar to Brownian motion on the Lie group $\mathrm{O}(n)$ as studied in [26]. Indeed the Brownian motion $B^{\mathrm{O}(n)}(t)$ on $\mathrm{O}(n)$ written as a Stratonovich SDE is of the form $dB^{\mathrm{O}(n)}(t) = \sum_{\ell<k}(B^{\mathrm{O}(n)}(t)E_{(\ell,k)}) \circ \mathrm{d}W^{(\ell,k)}(t)$. This similarity with Brownian motion on $\mathrm{O}(n)$ allows us to deduce a time reversal formula analogously to [26]. Indeed, denote by $\mathcal{X}(\mathrm{O}(n))$ the vector fields on $\mathrm{O}(n)$ and by $\nabla \ln p_t \in \mathcal{X}(\mathrm{O}(n))$ the gradient vector field of $\ln p_t$. We then need to transform the gradient vector field by the map $Q(s) : \mathcal{X}(\mathrm{O}(n)) \to \mathcal{X}(\mathrm{O}(n))$ given for $V \in \mathcal{X}(\mathrm{O}(n))$ and $X \in \mathcal{O}(n)$ as

$$(Q(s)V)(X) = \sum_{\ell<k} \frac{\langle V(X), XE_{(\ell,k)}\rangle}{(\lambda_k(t) - \lambda_\ell(t))^2} XE_{(\ell,k)}.$$

The resulting time reversal formula is for $Y(s) = X(T-s)$ given by

$$dY(s) = -\alpha(Q(T-s)\nabla \ln p_{T-s})(\bar{\lambda}, Y(s))\mathrm{d}s + \sqrt{\alpha}\sum_{\ell<k}\left(\frac{Y(s)E_{(\ell,k)}}{\lambda_k(T-s)-\lambda_\ell(T-s)}\right)\circ \mathrm{d}W^{(\ell,k)}(s). \tag{76}$$

In analogy to the forward dynamics above, this translates into an exponential update rule that in the notation of Eq. (3) (i.e., $T-s \to t$ and time-reversal denoted by an overline, $Y(s) \to \bar{X}(t)$) reads $\bar{X}(t+\mathrm{d}t) = \bar{X}(t\exp\big(\mathrm{d}\bar{Z}(t)\big)$ with $\mathrm{d}\bar{Z}(t) = -\alpha\nabla \ln p_t(\bar{\lambda}, \bar{X})\mathrm{d}t + \sqrt{\alpha}\sum_{\ell<k}\frac{E_{(\ell,k)}}{\lambda_k(t)-\lambda_\ell(t)}\mathrm{d}\bar{W}^{(\ell,k)}(t) \in \mathfrak{o}(n)$ (see also algorithm 1 in [26] for related results for Brownian dynamics on a manifold).

S.1 Preliminaries on SDEs on $\mathrm{O}(n)$

We first briefly recall some preliminary material on SDEs, particularly Stratonovich SDEs, on Lie groups. An excellent reference for this topic is [58], as well as the appendix in [26].

Vector Fields and Stratonovich SDEs on $\mathbb{R}^n$. Recall that a smooth vector field V on $\mathbb{R}^d$ is a map that smoothly assigns to each point a vector. Formally, we can therefore view V as a smooth map

$$V : \mathbb{R}^n \to \mathbb{R}^n, \qquad x \mapsto (V_1(x), \dots, V_n(x)).$$

Given two smooth vector fields V and U on $\mathbb{R}^d$, we define the covariant derivative $\nabla_U V$ of V with respect to U to be the vector field

$$\begin{aligned}(\nabla_U V)(x) &= \frac{d}{dt}\bigg|_{t=0} V(x+tU(x)) \\ &= \left(\sum_{j=1}^n U_j(x)\frac{\partial V_1}{\partial x_j}(x), \sum_{j=1}^n U_j(x)\frac{\partial V_2}{\partial x_j}(x), \dots, \sum_{j=1}^n U_j(x)\frac{\partial V_n}{\partial x_j}(x)\right).\end{aligned}$$

Given $d \geq 1$, denote by $W^1, \dots, W^d$ independent one-dimensional Brownian motions and let $V_1, \dots, V_d$ be vector fields on $\mathbb{R}^d$. Then the Stratonovich SDE

$$dX(t) = \sum_i V_i(X_t) \circ dW^i(t)$$

denotes the Ito SDE

$$dX(t) = \frac{1}{2}\left(\sum_i (\nabla_{V_i} V_i)(X(t))\right) dt + \sum_i V_i(X(t)) dW^i(t).$$

Just to have a concrete example in mind: The n-dimensional Brownian motion can be written by taking $d = n$ and considering the constant vector fields $V_i(x) = e_i$ for e_i the standard basis vectors.

Vector Fields and Stratonovich SDEs on $\mathrm{O}(n)$. In this subsection, we consider the concrete case of a Stratonovich SDE driven by left invariant vector fields on $\mathrm{O}(n)$. Indeed, a left invariant vector field V is determined by a Lie algebra element $z \in \mathfrak{o}(n)$ and given by

$$V(X) = Xz \in T_X \mathrm{O}(n).$$

We now exploit that $\mathrm{O}(n)$ is embedded in $\mathbb{R}^{n\times n}$ so we can write a Lie group SDE as an SDE on $\mathbb{R}^{n\times n}$. Indeed, let $V_1, \dots, V_d$ be left invariant vector fields $V_i(X) = Xz_i$ and consider the Lie group SDE

$$dX(t) = \sum_i (X(t)z_i) \circ dW^i(t).$$

Based on Section 1.2 in Hsu [58], we can write the above SDE as an Ito SDE on $\mathbb{R}^{n\times n}$ with the terms

$$dX(t) = \frac{1}{2}\sum_i (X(t)z_i^2) dt + \sum_i (X(t)z_i) dW^i(t). \tag{77}$$

S.2 Writing the Eigenvector SDE as a $\mathrm{O}(n)$ Stratonovich SDE

We now proof Proposition 4.

Proof. This follows by a direct calculation using equation (77) for time-dependent vector fields. Recall that $E_{(\ell,k)} = e_\ell e_k^T - e_k e_\ell^T$ for $\ell < k$, where e_i is the standard basis vector of $\mathbb{R}^n$ viewed as a row vector. Then denoting by e_{ii} the diagonal matrix that is 1 at the (i,i) entry and zero everywhere else, it holds that

$$E_{(\ell,k)}^2 = -(e_{\ell\ell} + e_{kk}).$$

Therefore the $\mathrm{O}(n)$ Stratonovich SDE from (75) is the Ito SDE on $\mathbb{R}^{n\times n}$ given by

$$dX(t) = -\frac{\alpha}{2}\sum_{\ell<k} \frac{X(t)(e_{\ell\ell} + e_{kk})}{(\lambda_k(t) - \lambda_\ell(t))^2} + \sqrt{\alpha}\sum_{\ell<k} \frac{X(t)E_{(\ell,k)}}{\lambda_k(t) - \lambda_\ell(t)} dW_{(\ell,k)}(t).$$

By writing $X(t) = (v_1(t) \dots v_n(t))$ as row vectors, the last equation is easily seen to be exactly the (Eigenvector-SDE). □

S.3 Deduction of Time Reversal Formula

Recall that the Brownian motion on $\mathrm{O}(n)$ is given by the $\mathrm{O}(n)$ Stratonovich SDE

$$dB^{\mathrm{O}(n)}(t) = \sum_{\ell<k} (B^{\mathrm{O}(n)}(t)E_{(\ell,k)}) \circ dW^{(l,k)}(t).$$

We next consider the time-scaled Brownian motion on $\mathrm{O}(n)$ given by

$$X(t) = g(t)B^{\mathrm{O}(n)}(t)$$

for some scalar function $g(t)$.

As in [26] we denote by $\nabla \ln p_{T-s}$ the gradient vector field of $\ln p_{T-s}$. It is shown (or rather stated that it can be shown) in Theorem 1 in De Bortoli et al. [26] that the time reversal of the Brownian motion $Y(s) = X(T-s)$ satisfies the SDE

$$dY(s) = g(T-s)^2 \nabla \ln p_{T-s}(Y(s))\, dt + g(T-s) dB^{\mathrm{O}(n)}(s).$$

The time reversal formula (76) is obtained by adapting the above result to our SDE (75), where the vector fields are scaled by the factors $\frac{\sqrt{\alpha}}{\lambda_k(t)-\lambda_\ell(t)}$. The final result is stated in (76).

S.4 Loss Function

As for the eigenvalue SDE, we have no direct access to the probability measures $p_t(x_t|x_0)$. We therefore employ a loss analogous to the one used to train the eigenvalues and explained in App. A.

To approximate the score, we observe that similarly to Equation (8) in De Bortoli et al. [26],

$$\lim_{t\to s}(t-s)(\nabla \ln p_{t|s})(X_t|X_s) \approx -\Sigma(t,s)\cdot \ln_{\mathrm{O}(n)}(X_t^T X_s) \quad \text{for} \quad \Sigma(t,s) = \operatorname{diag}\left(\frac{\alpha(t-s)}{(\lambda_k(s)-\lambda_\ell(s))^2}\right),$$

where $\ln_{\mathrm{O}(n)} : \mathrm{O}(n) \to \mathfrak{o}(n)$ is the matrix logarithm given for $X \in \mathrm{O}(n)$ by $\ln_G(X) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}(X-\mathrm{Id})^k$ and we view $\Sigma(t,s)$ as a diagonal matrix with respect to the standard basis $E_{(\ell,k)}$ of $\mathfrak{o}(n)$.

As discussed in the main part, for a small $h > 0$, we have the approximation

$$X(t+h) \approx X(t)\exp(Z) \qquad \text{with} \qquad Z = \sqrt{\alpha h}\sum_{\ell<k} \frac{E_{(\ell,k)}\mathcal{N}^{(\ell,k)}(0,1)}{\lambda_k(t)-\lambda_\ell(t)} \in \mathfrak{o}(n),$$

where $\mathcal{N}^{(\ell,k)}(0,1)$ are independent samples of standard 1-dimensional Gaussians and $\exp(Z) = \sum_{i=0}^{\infty}\frac{Z^i}{i!}$ is the matrix exponential.

Assume now that we have N discretizations of the interval $[0,T]$ for some $T>0$ that we denote for $1 \le r \le N$ as

$$0 = t_0^{(r)} < t_1^{(r)} < t_2^{(r)} < \ldots < t_{k^{(r)}-1}^{(r)} < t_{k^{(r)}}^{(r)} = T.$$

For those we have N sampled paths of our SDE from (75) denoted for $1 \le r \le N$ as

$$X_0^{(r)}, X_{t_1^{(r)}}^{(r)}, X_{t_2^{(r)}}^{(r)}, \ldots X_{t_{k^{(r)}-1}^{(r)}}^{(r)}, X_T^{(r)} \qquad \text{with Lie algebra increments} \qquad Z_0^{(r)}, Z_{t_1^{(r)}}^{(r)}, Z_{t_2^{(r)}}^{(r)} \ldots Z_{t_{k^{(r)}-1}^{(r)}}^{(r)}$$

so that for all $0 \le i \le k^{(r)}-1$ it holds that

$$X_{t_{i+1}^{(r)}}^{(r)} = X_{t_i^{(r)}}^{(r)} \exp\left(Z_{t_i^{(r)}}^{(r)}\right) \qquad \text{and therefore} \qquad -Z_{t_i^{(r)}}^{(r)} = \ln_{\mathrm{O}(n)}((X_{t_{i+1}^{(r)}}^{(r)})^T X_{t_i^{(r)}}^{(r)}).$$

By the last equation, it therefore follows that

$$\nabla \ln p_{t|s}(X_{t_{i+1}^{(r)}}^{(r)}|X_{t_i^{(r)}}^{(r)}) \approx \frac{\Sigma(t_{i+1}^{(r)}, t_i^{(r)})Z_{t_i^{(r)}}^{(r)}}{t_{i+1}^{(r)} - t_i^{(r)}}.$$

Analogously to App. A, we therefore want to optimize the loss

$$\frac{1}{N}\sum_{r=1}^{N}\sum_{i=1}^{k^{(r)}-1} \frac{t_{i+1}^{(r)}-t_i^{(r)}}{T}\left|\left|s(X_{t_{i+1}^{(r)}}^{(r)}, t_{i+1}^{(r)}) - \frac{\Sigma(t_{i+1}^{(r)}, t_i^{(r)})Z_{t_i^{(r)}}^{(r)}}{t_{i+1}^{(r)}-t_i^{(r)}}\right|\right|,$$

where the norm is the L^2-norm on the Lie algebra $\mathfrak{o}(n)$.

S.5 Invariance on the Lie algebra

Since permutations of adjacency matrices have the same spectrum, the permutation invariance of graphs did not have to be considered when learning purely on spectral information, see, e.g., Fig. 1. This changes for the eigenvectors in $X(t) = (v_1(t), \ldots, v_n(t)) \in \mathrm{O}(n)$. While the column indices are fixed since each eigenvector $v_i(t)$ corresponds to a fixed eigenvalue $\lambda_i(t)$, the row indices change upon permutation of the adjacency matrix. Therefore, the desired learning algorithm must be invariant under permutations $X \mapsto PX$, $Y \mapsto PY$, where P is a permutation matrix swapping the rows of X and Y, respectively. The backwards dynamics is mathematically described and numerically implemented as $Y(t-\mathrm{d}t) = Y(t)\exp(\mathrm{d}Z_Y(\lambda, Y, t))$, which under permutation $Y \mapsto PY$ reads $PY(t-\mathrm{d}t) = PY(t)\exp(\mathrm{d}Z_Y(\lambda, PY, t))$. At the same time, multiplication of the former equation from the left side by P implies $PY(t-\mathrm{d}t) = PY(t)\exp(\mathrm{d}Z_Y(\lambda, Y, t))$. Since PY is invertible, these equations imply $\exp(\mathrm{d}Z_Y(\lambda, Y, t)) = \exp(\mathrm{d}Z_Y(\lambda, PY, t))$. Since $\mathrm{d}Z_Y(\lambda, Y, t) \to 0$ as

$\mathrm{d}t \to 0$, we can deduce that $\mathrm{d}Z_Y(\lambda, Y, t) = \mathrm{d}Z_Y(\lambda, PY, t)$, i.e., that the increment in the exponent (meaning the change phrased in terms of the Lie algebra) is permutation invariant. Since we learn the score on these Lie algebra increments, see Eq. (76), this implies that the score is invariant under permutations $Y \to PY$. Crucially, this differs from the typical time reversal score as in Eq. (2), which would be permutation equivariant, $s(PYP^T, t) = Ps(Y, t)P^T$ (note that on the full matrix, the permutation changes rows and columns, i.e., $Y \to PYP^T$). We instead have invariance $s(PY, t) = s(Y, t)$, and no equivariance $s(PY, t) \neq Ps(Y, t)$. Note that this somewhat unexpected absence of equivariance is due to the choice to parametrize the score in the Lie algebra instead of the Lie group which is seen from the form $Y(t - \mathrm{d}t) = Y(t) \exp(\mathrm{d}Z_Y(\lambda, Y, t))$. Here, when updating Y in a permuted frame, all the information about the permutation is already contained in the $Y(t)$ which enables that the exponent (containing the score) is unaffected by the permutation. This invariance opens the learning task to Set Transformers.

S.6 Sign symmetry

We now show that the score obeys a sign symmetry law, which allows to reduce the space of the learning target by a factor of 2^d. In particular, we show that the score obeys $\nabla \ln p_t(\bar{\lambda}, \bar{X}_t F) = F \nabla \ln p_t(\bar{\lambda}, \bar{X}_t) F$ for any diagonal matrix F with entries ± 1 representing sign flips.

To see this, compare the sign-flipped dynamics $\bar{X}_{t+\mathrm{d}t} F = \bar{X}_t F \exp\big(\mathrm{d}\bar{Z}_t(\lambda, \bar{X}_t F)\big)$, to the original dynamics multiplied by F, i.e., $\bar{X}_{t+\mathrm{d}t} F = \bar{X}_t \exp\big(\mathrm{d}\bar{Z}_t(\lambda, \bar{X}_t)\big) F$. Using $F = F^{-1}$, the latter yields $\bar{X}_{t+\mathrm{d}t} F = \bar{X}_t F \exp\big(F \mathrm{d}\bar{Z}_t(\lambda, \bar{X}_t) F\big)$, which since $\bar{X}_t F$ is invertible and $\exp$ can be linearized for small increments, implies $\mathrm{d}\bar{Z}_t(\lambda, \bar{X}_t F) = F \mathrm{d}\bar{Z}_t(\lambda, \bar{X}_t) F$. Since Wiener increments are symmetric in the sign, this is equivalent to $\nabla \ln p_t(\bar{\lambda}, \bar{X}_t F) = F \nabla \ln p_t(\bar{\lambda}, \bar{X}_t) F$.

Thus, the learning target is effectively reduced by a factor of 2^d – i.e. an exponential reduction in the dimension. See App. S.9 for a clear practical discussion on how this can be precisely exploited algorithmically.

S.7 Preservation of Orthonormality property under Simplification

As described in Section 3.3, dropping the λ dependence in the Eigenvector forward dynamics retains the desired Orthonormality property. In particular, changing the incremental update of Eq. 6 to

$$\mathrm{d}Z(t) = \sqrt{\alpha} \sum_{\ell < k} E_{(\ell,k)} \mathrm{d}W^{(l,k)}(t),$$

still gives an increment in the Lie algebra $\mathfrak{o}(n)$ and thus retains the dynamics on the Lie group $O(n)$. The time reversal thereby changes to

$$\mathrm{d}\bar{Z}(t) = \alpha \nabla \ln p_t(\bar{\lambda}, \bar{X}) \mathrm{d}t + \sqrt{\alpha} \sum_{\ell < k} E_{(\ell,k)} \mathrm{d}\bar{W}^{(\ell,k)}(t) \in \mathfrak{o}(n).$$

Note that the score *remains* dependent on the eigenvalues, since in the data distribution, eigenvalues and eigenvectors are generally correlated. Thus, even though the propagation is now simplified, the score network for the eigenvectors needs the spectrum as input. Further, note that the invariant distribution of the eigenvectors remains the uniform distribution on $O(n)$ up to the sign of the determinant, since rotation invariance of the dynamics is preserved.

S.8 Handling non-isomorphic but isospectral graphs

Non-isomorphic but isospectral graphs are only a small share in the data we considered here (e.g. in Community Small, we found no non-isomorphic cospectral graphs, while in our Brain training set, 28 of 10,500 graph instances, or 0.27%, are non-isomprohic but isospectral). Thus, they are very rare. Further, as we explain below, they do *not* pose a challenge to our architecture.

From a theoretical perspective, note that our diffusion model learns graphs by (1) learning the spectra, let's call it $p_{\text{spec}}(\lambda)$, and by learning the conditioned eigenvectors, say $p_{\text{eigvec}}(X \mid \lambda)$. A graph is fully described by its spectrum λ and the tuple of eigenvectors X, which can be decomposed into: $p(\lambda, X) = p_{\text{spec}}(\lambda) p_{\text{eigvec}}(X \mid \lambda)$. Even though the spectrum λ is identical for (the few) isospectral but non-isomorphic graphs, the eigenvectors X will be different, thus sampling from the eigenvector SDE resolves this ambiguity.

In addition, we constructed a simple experiment, where our model should learn to distinguish two of the isospectral but non-isomorphic graphs from the Brain dataset, say graph G and graph H. Our training set consists of: 50% graph G, 50% graph H. We trained (without hyperparameter tuning for this illustrative run) and obtained during sampling in all cases (i.e., 100%) the same spectrum, as expected, but combined with the eigenvector sampling, we obtained graphs which are in our small experiment: 54% isomorphic to G, 46% isomorphic to graph H, and 0% isomorphic to neither. Thus, the model can distinguish the graphs even though they are isospectral. Although these graphs are rare, our model can thus, also empirically, adequately handle them.

S.9 Numerical illustration

As proven in Lemma 5, we open the learning problem to our best knowledge for the first time to the field of permutation *in*variant architectures, such as Set transformers [30]. In particular, we train a set transformer to learn the score of the time reversal described in 3.3, and present the results in Table 7. We also illustrate these results visually by showing in Fig. 12 explicitly the generated graphs. Note that generation of $n = 1000$ such graphs takes only 17 seconds.

Eigenvector score architecture. For the conditional eigenvector score learning of the Lie Algebra we use a row-invariance Set transformer [30]. The model takes as input a diffusion time t, the Laplacian spectrum excluding the fixed first Laplacian eigenvalue, and an eigenvector matrix $X \in O(d)$. The output is a vector of strict upper-triangular Lie algebra coordinates in $\mathfrak{o}(d)$, corresponding to the $d(d-1)/2$ pairwise eigenvector rotation directions.

We treat the rows of X as an unordered set of tokens. Each row is embedded into a width-512 token representation. The time variable is encoded using Fourier features with embedding dimension 256 and frequency range $[1, 100]$. The spectrum and time embeddings are combined into a width-512 conditioning vector. The row-token trunk consists of three conditioned self-attention blocks with 16 attention heads. Conditioning is injected by FiLM-style affine modulation [59] before both the attention and token-MLP sublayers. We use scaled SiLU activations.

For every pair $k < \ell$, we form pair features from C_k, C_ℓ, their difference, absolute difference, elementwise product, the corresponding spectral features

$$\lambda_k, \quad \lambda_\ell, \quad \lambda_k - \lambda_\ell, \quad |\lambda_k - \lambda_\ell|,$$

and a global context vector. The global context is computed from the conditioning vector and from row-token mean, variance, and smooth-max pooled statistics. A width-512 pair MLP of depth 1 maps each pair feature to the corresponding Lie algebra coordinate.

To apply our proven sign symmetry (see App. S.6), we proceed as follows: Eigenvector column signs are first put into a deterministic canonical gauge, the scalar pair head is evaluated in that gauge, and the output is transformed back to the original gauge using the corresponding sign flips, as detailed in App. S.6. This enforces the correct equivariance under eigenvector sign flips while preserving the expressivity of the unconstrained pair head, and reduces the learning problem by a factor of 2^d (see App. S.6).

As for the spectra, we use EMA (exponential moving averages) on the trained models.

Training setup. We train on the time interval $[0, T]$ and use a fixed learning grid with base spacing $\mathrm{d}t = 0.05$ and refinement factors

time interval	factor
$[0, 0.25]$	0.03125
$[0.25, 0.5]$	0.0625
$[0.5, 1]$	0.125
$[1, 2]$	0.25
$[2, 3]$	0.5
$[3, 8]$	1.0

Thus, the earliest grid spacing is 0.0015625. We simulate 4000 forward paths per epoch and use all fixed-grid intervals in the eigenvector loss. The training batch size is 10, and the model is trained for 80 epochs. The reported model in Table 7 used EMA with $\alpha = 0.95$ and averaged over the models from epochs 78 to 80.

Table 6: Comparison of graph statistics between DyDM and established graph generative models using maximum mean discrepancy (MMD) distance, on Community Small. Lower values indicate better performance. Best results are highlighted in bold.

Model	Degree	Clustering	Orbit	Spectral	Mean
DyDM (ours)	**0.0002**	**0.0031**	**0.0021**	**0.0011**	**0.0016**
EDP-GNN	0.0270	0.0741	0.0170	0.0280	0.0365
GDSS	0.0057	0.0755	0.0232	0.0340	0.0346
ConGress	0.0025	0.0735	0.0042	0.0279	0.0270
DiGress (no trick)	0.0117	0.0796	0.4308	0.0420	0.1410
DiGress (trick)	**0.0001**	**0.0052**	**0.0034**	**0.0024**	**0.0028**

Table 7: MMD metrics of generated graphs trained on the WL-Bimodal dataset. Consistently with the WL-blindness of GNNs, EDP-GNN is very good at learning the WL-equivalence class of 3-regular graphs (degree distance is 0), but gets outperformed by the DyDM + Lie group diffusion approach on the other distances. Top table: Average of multiple training runs, bottom table: empirical standard deviation for completeness.

Mean Distances	Degree	Clustering Coefficient	Spectral	Orbit
Lie Diffusion Eigenvectors	0.00013	**0.00041**	**0.00011**	**0.00001**
EDP-GNN	**0.00000**	0.16590	0.05011	0.00189
GDSS	**0.00000**	0.33342	0.12303	0.00025
ConGress	0.07759	0.45267	0.15022	0.01024
DiGress (no trick)	0.37712	0.70940	0.23426	0.04024
DiGress (trick)	0.00000	**0.00189**	**0.00050**	**0.00004**

Empirical Standard Deviations	Degree	Clustering Coefficient	Spectral	Orbit
Lie Diffusion Eigenvectors	6.3e-05	6.5e-05	4.1e-05	9.7e-06
EDP-GNN	6e-08	0.11	0.035	0.0007
GDSS	6.5e-08	0.028	0.009	0.00011
ConGress	0.17	0.17	0.053	0.018
DiGress (no trick)	0.004	0.0023	0.0028	0.011
DiGress (trick)	3.3e-07	0.0016	0.00042	2.6e-05

Optimization uses AdamW through Optax. For the eigenvector model, we use global gradient clipping at norm 0.5, initial learning rate $7.5 \cdot 10^{-4}$, final learning rate $1.2 \cdot 10^{-4}$, weight decay 10^{-5}, and a warmup fraction of 0.07 with a cosine decay schedule.

S.9.1 Compute Resources

On one $A100$ gpu, sampling of 1000 graphs from our model takes 17 seconds. Training of the eigenvector model takes 20 minutes. We trained in total 5 models with different seeds and report their average and empirical standard deviation in Table 7. Consistent with our discussion of the challenges that WL-equivalent graphs pose to GNNs, we see that models like EDP-GNN are very good at learning the degree, since all graphs in the same equivalence class have the same degree distribution. Indeed, all graphs of the WL-Bimodal dataset have the same degree distribution. However, distances which capture more complexity of the graph like Clustering Coefficient, Spectral or Orbit MMD distance show clearly the advantage of our proposed spectrum-first method, and they show evidently that our Lie group diffusion learns accurately. We depict in Fig. 12 graphs that were generated using the Lie group diffusion.

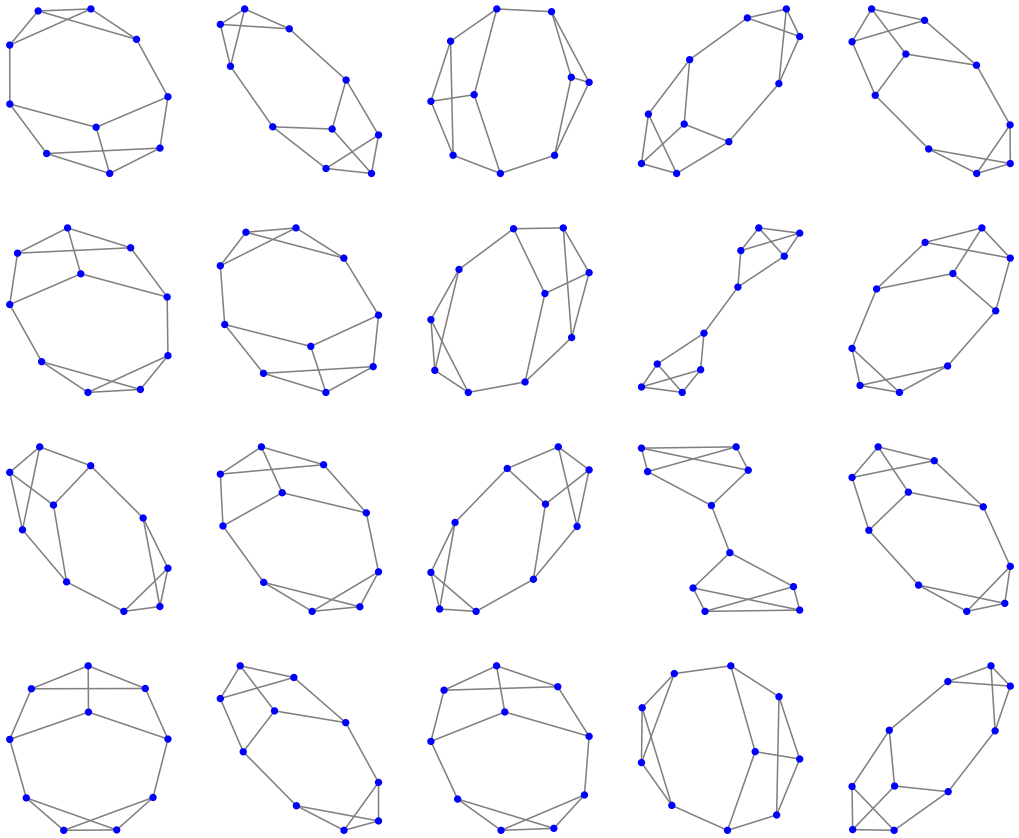


Figure 12: Graphs generated using the Lie Group diffusion: Each generated graph is isomorphic to one of the WL equivalent graphs, and as evident from Table 7, it learns the distribution very well. Sampling of 1,000 such graphs takes 17 seconds on an A100 GPU.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly outline our contributions with bullet points in the introduction (Section 1). We provide proofs in the Appendix for the multiple theoretical contributions we make and provide numerical evidence through experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations throughout. Moreover, we include a separate section explicitly pointing out limitations (Section 6) and pointing towards ways to overcome them, derive the computational complexity of our developed methods (e.g. Section H), and report explicitly runtime for both our methods (e.g. in Section K and Section S.9.1) and other benchmarked methods (e.g. in Section Q.2)

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Assumptions are clearly stated and Theorems and Lemmata are proven in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide in the uploaded supplemental material all data that is needed to run the experiments, including datasets. We also include the generated output from all models in the Supplemental Material. Further, we describe in the Appendix in detail how the experiments were conducted.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We include in the uploaded supplemental material all datasets, and the generated samples for each model that we train, i.e. including ours and including data generated from other models. We describe our model in much detail in the Appendix, to aid implementation. We plan to release the implementation publicly at publication at github.com/schwarzTass/DysonDiffusionModel.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Even though the main contribution of this paper is not a novel architecture, we still explain the design, type of optimizer (AdamW), and hyperparameter in the Appendix (e.g. in Section K). We provide a critical in-depth discussion of the datasets in Section P, where we also detail the data splits, and follow recent guidance on benchmarking in the field [32].

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: To ensure statistical robustness, we sample for the results on the spectral sampling for each run 1000 times and, in addition to the mean, report the Wasserstein distance of the marginals (see Section 4.2 and last part of Section K). Furthermore, we report wherever feasible standard deviations over multiple runs where the seed value and hence the training was changed. For the spectral samples, we report all run outputs in the uploaded supplemental material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report this in the corresponding Appendix of each experiment and model (e.g. Section Q.2 or Section S.9.1).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research is of foundational nature and conforms with the Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our research is of foundational nature and we do not see a direct negative societal impact that could arise if applied and misused, beyond general risks of AI.

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not believe that this paper has high risk of misuse.

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: We provide references to every source of data, and we provide references to the code packages we implement the model in Section K.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We introduce the WL-Bimodal dataset, which is documented in Section P and depicted in Figure 2, as well as provided in the uploaded supplemental material.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: This paper contains neither research with human subjects nor crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper contains neither research with human subjects nor crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We used LLMs to aid with standard implementations, which we acknowledge explicitly.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.