Graph-Aware Diffusion for Signal Generation

Vimal Kumarasamy Balasubramanian[†]

[†] Delft University of Technology, The Netherlands

IEEE ICASSP 2026

Generating signals on a graph

- **Generative modeling:** learn to draw new samples from an unknown distribution p_0, given only samples $\{\mathbf{x}_0^{(i)}\}_{i=1}^{M} \sim p_0$

  $\Rightarrow$ huge success in images, audio, and *graph* generation

Generating signals on a graph

- **Generative modeling:** learn to draw new samples from an unknown distribution p_0, given only samples $\{\mathbf{x}_0^{(i)}\}_{i=1}^{M} \sim p_0$
  - $\Rightarrow$ huge success in images, audio, and *graph* generation

- Far less studied: generating graph signals on a known, fixed graph $\mathcal{G}$
  - $\Rightarrow$ a signal $\mathbf{x} \in \mathbb{R}^N$ is a value per node (sensor reading, rating, speed, ...)
  - $\Rightarrow$ most graph-diffusion work generates *the graph*; here the graph is given

Generating signals on a graph

- **Generative modeling:** learn to draw new samples from an unknown distribution p_0, given only samples $\{\mathbf{x}_0^{(i)}\}_{i=1}^{M} \sim p_0$
  - $\Rightarrow$ huge success in images, audio, and *graph* generation

- Far less studied: generating graph signals on a known, fixed graph $\mathcal{G}$
  - $\Rightarrow$ a signal $\mathbf{x} \in \mathbb{R}^N$ is a value per node (sensor reading, rating, speed, ...)
  - $\Rightarrow$ most graph-diffusion work generates *the graph*; here the graph is given

- **Why exploit a known graph?** It is a strong, free prior:
  - $\Rightarrow$ bake the structure into both the forward *and* the backward process
  - $\Rightarrow$ reuse the GSP toolbox – graph filters, smoothness priors, the spectrum
  - $\Rightarrow$ generated signals respect the topology by construction

Generating signals on a graph

- **Generative modeling:** learn to draw new samples from an unknown distribution p_0, given only samples $\{\mathbf{x}_0^{(i)}\}_{i=1}^{M} \sim p_0$
  - $\Rightarrow$ huge success in images, audio, and *graph* generation

- Far less studied: generating graph signals on a known, fixed graph $\mathcal{G}$
  - $\Rightarrow$ a signal $\mathbf{x} \in \mathbb{R}^N$ is a value per node (sensor reading, rating, speed, ...)
  - $\Rightarrow$ most graph-diffusion work generates *the graph*; here the graph is given

- **Why exploit a known graph?** It is a strong, free prior:
  - $\Rightarrow$ bake the structure into both the forward *and* the backward process
  - $\Rightarrow$ reuse the GSP toolbox – graph filters, smoothness priors, the spectrum
  - $\Rightarrow$ generated signals respect the topology by construction

- **Applications:** probabilistic forecasting, recommender systems, data imputation in sensor / traffic networks, wireless, 3D meshes ...

Diffusion models: a generic SDE framework

- A forward SDE gradually turns data into noise; a backward SDE turns noise back into data

Forward (data → noise)

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)\, dt + g(\mathbf{x}_t, t)\, d\mathbf{w}_t$$

- drift f, diffusion g, Brownian motion $\mathbf{w}_t$
  ⇒ converges to a simple distribution p_T

Backward (noise → data)

$$d\mathbf{x}_t = \big[\, f(\mathbf{x}_t, t) - g(\mathbf{x}_t, t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \,\big] dt + g\, d\mathbf{w}_t$$

- runs in reverse time from a sample of p_T
  ⇒ needs only the score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$

Diffusion models: a generic SDE framework

- A forward SDE gradually turns data into noise; a backward SDE turns noise back into data

Forward (data → noise)

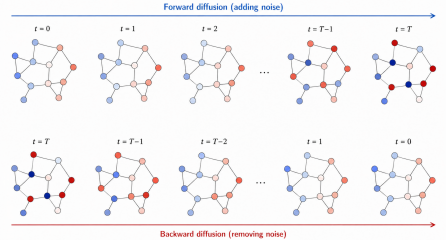
$$d\mathbf{x}_t = f(\mathbf{x}_t, t)\, dt + g(\mathbf{x}_t, t)\, d\mathbf{w}_t$$

- drift f, diffusion g, Brownian motion $\mathbf{w}_t$
  - $\Rightarrow$ converges to a simple distribution p_T

Backward (noise → data)

$$d\mathbf{x}_t = \left[f(\mathbf{x}_t, t) - g(\mathbf{x}_t, t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \right] dt + g\, d\mathbf{w}_t$$

- runs in reverse time from a sample of p_T
  - $\Rightarrow$ needs only the score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$



- **Our novelty:** particularize this framework to a known graph $\Rightarrow$ graph-aware forward *and* backward

GAD: put the graph in the drift

- We use the heat equation on the Laplacian – a (time-inhomogeneous) Ornstein–Uhlenbeck SDE

$$d\mathbf{x}_t = -\,c_t\,\mathbf{L}_\gamma\,\mathbf{x}_t\,dt + \sqrt{2c_t}\,\sigma\,d\mathbf{w}_t, \qquad \mathbf{L}_\gamma = \mathbf{L} + \gamma\mathbf{I}, \quad \gamma > 0.$$

Unlike most works, the graph enters the drift – not only the denoiser.

GAD: put the graph in the drift

- We use the heat equation on the Laplacian – a (time-inhomogeneous) Ornstein–Uhlenbeck SDE

$$d\mathbf{x}_t = -\,c_t\,\mathbf{L}_\gamma\,\mathbf{x}_t\,dt + \sqrt{2c_t}\,\sigma\,d\mathbf{w}_t, \qquad \mathbf{L}_\gamma = \mathbf{L} + \gamma\mathbf{I}, \quad \gamma > 0.$$

Unlike most works, the graph enters the drift – not only the denoiser.

- The marginals are available in closed form (a GMRF):

$$\boldsymbol{\mu}_t = \mathbf{H}_t\,\mathbf{x}_0, \qquad \boldsymbol{\Sigma}_t = \sigma^2\big(\mathbf{I} - \mathbf{H}_t^2\big)\mathbf{L}_\gamma^{-1}, \qquad \mathbf{H}_t = e^{-\bar{c}_t\mathbf{L}_\gamma}, \quad \bar{c}_t = \int_0^t c_s\,ds.$$

$\mathbf{H}_t$ is a low-pass graph filter that attenuates the signal and injects graph-structured noise.

GAD: put the graph in the drift

- We use the heat equation on the Laplacian – a (time-inhomogeneous) Ornstein–Uhlenbeck SDE

$$d\mathbf{x}_t = -\,c_t\,\mathbf{L}_\gamma\,\mathbf{x}_t\,dt + \sqrt{2c_t}\,\sigma\,d\mathbf{w}_t, \qquad \mathbf{L}_\gamma = \mathbf{L} + \gamma\mathbf{I}, \quad \gamma > 0.$$

Unlike most works, the graph enters the drift – not only the denoiser.

- The marginals are available in closed form (a GMRF):

$$\boldsymbol{\mu}_t = \mathbf{H}_t\,\mathbf{x}_0, \qquad \boldsymbol{\Sigma}_t = \sigma^2\big(\mathbf{I} - \mathbf{H}_t^2\big)\mathbf{L}_\gamma^{-1}, \qquad \mathbf{H}_t = e^{-\bar{c}_t\mathbf{L}_\gamma}, \quad \bar{c}_t = \int_0^t c_s\,ds.$$

$\mathbf{H}_t$ is a low-pass graph filter that attenuates the signal and injects graph-structured noise.

- **Convergence:** as $\bar{c}_t \to \infty$ the process reaches a graph-dependent stationary distribution

$$p_\infty = \mathcal{N}\big(\mathbf{0},\ \sigma^2\mathbf{L}_\gamma^{-1}\big) \qquad \Longrightarrow \qquad \text{the simple end distribution is shaped by the graph.}$$

γ makes $-c_t\mathbf{L}_\gamma$ Hurwitz (guarantees convergence); too small → slow modes, too large → geometry lost.

The scheduler controls how fast we noise

- $\mathbf{H}_t = e^{-\bar{c}_t \mathbf{L}^\gamma}$ decays exponentially. With a uniform linear scheduler, dominant graph modes are noised too early $\Rightarrow$ poor reverse generation.

The scheduler controls how fast we noise

- $\mathbf{H}_t = e^{-\bar{c}_t \mathbf{L}_\gamma}$ decays exponentially. With a uniform linear scheduler, dominant graph modes are noised too early $\Rightarrow$ poor reverse generation.

- We use a floor-constrained polynomial scheduler (FCPS) to slow down the early decay:

$$c_t = c_{\min} + k u^\alpha, \qquad \bar{c}_t = c_{\min} t + (c_0 - c_{\min} T) u^{\alpha+1}, \qquad u = t/T.$$

k is chosen such that $\bar{c}_T = c_0$. For $\alpha \gg 1$, the term $u^{\alpha+1}$ grows slowly near $t = 0$, so graph modes are noised more gradually.

The scheduler controls how fast we noise

- $\mathbf{H}_t = e^{-\bar{c}_t \mathbf{L}^\gamma}$ decays exponentially. With a uniform linear scheduler, dominant graph modes are noised too early $\Rightarrow$ poor reverse generation.

- We use a floor-constrained polynomial scheduler (FCPS) to slow down the early decay:

$$c_t = c_{\min} + k u^\alpha, \qquad \bar{c}_t = c_{\min} t + (c_0 - c_{\min} T) u^{\alpha+1}, \qquad u = t/T.$$

k is chosen such that $\bar{c}_T = c_0$. For $\alpha \gg 1$, the term $u^{\alpha+1}$ grows slowly near $t = 0$, so graph modes are noised more gradually.

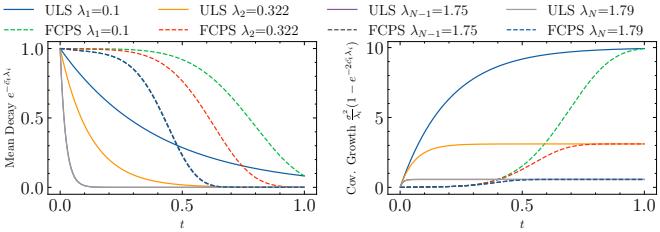


Figure. FCPS delays mean decay and covariance growth compared with ULS.

Backward process: the score is a denoiser

- The corresponding backward SDE keeps the graph in the drift:

$$d\mathbf{x}_t = -c_t \mathbf{L}_\gamma \mathbf{x}_t \, dt \; - \; 2c_t \sigma^2 \, \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \, dt + \sqrt{2c_t} \, \sigma \, d\mathbf{w}_t .$$

Backward process: the score is a denoiser

- The corresponding backward SDE keeps the graph in the drift:

$$d\mathbf{x}_t = -c_t \mathbf{L}_\gamma \mathbf{x}_t \, dt \; - \; 2c_t \sigma^2 \, \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \, dt + \sqrt{2c_t} \, \sigma \, d\mathbf{w}_t .$$

- Tweedie's formula turns score estimation into denoising:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \boldsymbol{\Sigma}_t^{-1} (\mathbf{H}_t \, \mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t] - \mathbf{x}_t) ,$$

$$\mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t] = g^\star(\mathbf{x}_t), \qquad g^\star \in \operatorname*{argmin}_g \mathbb{E}\left[\|\mathbf{x}_0 - g(\mathbf{x}_t)\|^2 \right] .$$

Since $\mathbf{H}_t$ and $\boldsymbol{\Sigma}_t$ are known, the hard part is learning the clean-signal predictor $g^\star$.

Backward process: the score is a denoiser

- The corresponding backward SDE keeps the graph in the drift:

$$d\mathbf{x}_t = -c_t \mathbf{L}_\gamma \mathbf{x}_t \, dt \; - \; 2c_t \sigma^2 \, \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \, dt + \sqrt{2c_t} \, \sigma \, d\mathbf{w}_t .$$

- Tweedie's formula turns score estimation into denoising:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \boldsymbol{\Sigma}_t^{-1} (\mathbf{H}_t \, \mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t] - \mathbf{x}_t) ,$$

$$\mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t] = g^\star(\mathbf{x}_t), \qquad g^\star \in \operatorname*{argmin}_g \mathbb{E}\big[\|\mathbf{x}_0 - g(\mathbf{x}_t)\|^2\big] .$$

Since $\mathbf{H}_t$ and $\boldsymbol{\Sigma}_t$ are known, the hard part is learning the clean-signal predictor $g^\star$.

- We use a GCNN denoiser: a cascade of learnable graph filters, conditioned on t:

$$\hat{\mathbf{x}}_0 = g_{\boldsymbol{\Theta}}(\mathbf{x}_t, \mathbf{L}, t), \qquad \boldsymbol{\Theta}^\star \in \operatorname*{argmin}_{\boldsymbol{\Theta}} \; \mathbb{E}\big[\|\mathbf{x}_0 - g_{\boldsymbol{\Theta}}(\mathbf{x}_t, \mathbf{L}, t)\|^2\big] .$$

Graph filters mix information through neighborhoods; conditioning on t lets the denoiser adapt from coarse recovery to fine-detail reconstruction.

Algorithm: training and sampling GAD

Algorithm 1: Training of GAD

Input: $\mathbf{L}$, data $\mathcal{D}$, T, c_t, σ, lr η.
Output: trained denoiser $g_{\boldsymbol{\Theta}^\star}$.

1. Sample a clean signal and time:
$$\mathbf{x}_0 \sim \mathcal{D}, \qquad t \sim \mathcal{U}(0, T).$$
2. Compute the forward marginal:
$$\mathbf{L}_\gamma = \mathbf{L} + \gamma \mathbf{I}, \qquad \mathbf{H}_t = e^{-\bar{c}_t \mathbf{L}_\gamma},$$
$$\boldsymbol{\mu}_t = \mathbf{H}_t \mathbf{x}_0, \qquad \boldsymbol{\Sigma}_t = \sigma^2 (\mathbf{I} - \mathbf{H}_t^2) \mathbf{L}_\gamma^{-1}.$$
3. Corrupt and denoise:
$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \qquad \hat{\mathbf{x}}_0 = g_{\boldsymbol{\Theta}}(\mathbf{x}_t, \mathbf{L}, t).$$
4. Update the GCNN denoiser:
$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \eta \nabla_{\boldsymbol{\Theta}} \|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|_2^2.$$

Algorithm: training and sampling GAD

Algorithm 1: Training of GAD

Input: $\mathbf{L}$, data $\mathcal{D}$, T, c_t, σ, lr η.
Output: trained denoiser $g_{\boldsymbol{\Theta}^\star}$.

1. Sample a clean signal and time:
$$\mathbf{x}_0 \sim \mathcal{D}, \qquad t \sim \mathcal{U}(0, T).$$
2. Compute the forward marginal:
$$\mathbf{L}_\gamma = \mathbf{L} + \gamma \mathbf{I}, \qquad \mathbf{H}_t = e^{-\bar{c}_t \mathbf{L}_\gamma},$$
$$\boldsymbol{\mu}_t = \mathbf{H}_t \mathbf{x}_0, \qquad \boldsymbol{\Sigma}_t = \sigma^2 (\mathbf{I} - \mathbf{H}_t^2) \mathbf{L}_\gamma^{-1}.$$
3. Corrupt and denoise:
$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \qquad \hat{\mathbf{x}}_0 = g_{\boldsymbol{\Theta}}(\mathbf{x}_t, \mathbf{L}, t).$$
4. Update the GCNN denoiser:
$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \eta \nabla_{\boldsymbol{\Theta}} \|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|_2^2.$$

Algorithm 2: Sampling of GAD

Input: $\mathbf{L}_\gamma$, σ, grid $0 = t_0 < \cdots < t_N = T$, $g_{\boldsymbol{\Theta}^\star}$.
Output: generated signal $\mathbf{x}_{t_0}$.

1. Initialize from the terminal prior:
$$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{L}_\gamma^{-1}).$$
2. For $i = N, \ldots, 1$, set
$$t = t_i, \qquad t^- = t_{i-1}, \qquad \Delta t = t - t^-.$$
3. Estimate $\mathbf{x}_0$ and the score:
$$\hat{\mathbf{x}}_0 = g_{\boldsymbol{\Theta}^\star}(\mathbf{x}_t, \mathbf{L}, t),$$
$$\hat{\mathbf{s}}_t = \boldsymbol{\Sigma}_t^{-1} (\mathbf{H}_t \hat{\mathbf{x}}_0 - \mathbf{x}_t).$$
4. Reverse Euler–Maruyama step:
$$\mathbf{x}_{t^-} \leftarrow \mathbf{x}_t + \left(c_t \mathbf{L}_\gamma \mathbf{x}_t + 2 c_t \sigma^2 \hat{\mathbf{s}}_t\right) \Delta t + \sqrt{2 c_t}\, \sigma \sqrt{\Delta t}\, \mathbf{z}_i, \quad \mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$
5. Return $\mathbf{x}_{t_0}$.

A GSP view: the optimal denoiser is a graph filter

- To build intuition, look at the denoising problem for a single signal at a fixed t. The noise model is fully known:

$$\mathbf{x}_t = \mathbf{H}_t \mathbf{x}_0 + \mathbf{e}, \qquad \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t), \qquad \mathbf{H}_t, \mathbf{\Sigma}_t \text{ known}.$$

A GSP view: the optimal denoiser is a graph filter

- To build intuition, look at the denoising problem for a single signal at a fixed t. The noise model is fully known:

$$\mathbf{x}_t = \mathbf{H}_t \mathbf{x}_0 + \mathbf{e}, \qquad \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t), \qquad \mathbf{H}_t, \mathbf{\Sigma}_t \text{ known}.$$

- Many graph signals are smooth, so we add a Laplacian smoothing prior:

$$\hat{\mathbf{x}}_0 = g(\mathbf{x}_t) = \underset{\mathbf{x}_0}{\operatorname{argmin}} \ \|\mathbf{x}_t - \mathbf{H}_t \mathbf{x}_0\|^2_{\mathbf{\Sigma}_t^{-1}} + \alpha \|\mathbf{x}_0\|^2_{\mathbf{L}}.$$

A GSP view: the optimal denoiser is a graph filter

- To build intuition, look at the denoising problem for a single signal at a fixed t. The noise model is fully known:

$$\mathbf{x}_t = \mathbf{H}_t \mathbf{x}_0 + \mathbf{e}, \qquad \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t), \qquad \mathbf{H}_t, \mathbf{\Sigma}_t \text{ known}.$$

- Many graph signals are smooth, so we add a Laplacian smoothing prior:

$$\hat{\mathbf{x}}_0 = g(\mathbf{x}_t) = \underset{\mathbf{x}_0}{\operatorname{argmin}} \ \|\mathbf{x}_t - \mathbf{H}_t \mathbf{x}_0\|^2_{\mathbf{\Sigma}_t^{-1}} + \alpha \|\mathbf{x}_0\|^2_{\mathbf{L}}.$$

Proposition: the solution is a graph filter

The minimizer is $\hat{\mathbf{x}}_0 = h(\mathbf{L})\mathbf{x}_t$, an ARMA graph filter.

$$h(\mathbf{L}) = \left(\mathbf{H}_t^\top \mathbf{\Sigma}_t^{-1} \mathbf{H}_t + \alpha \mathbf{L}\right)^{-1} \mathbf{H}_t^\top \mathbf{\Sigma}_t^{-1},$$

$$h(\lambda) = \frac{e^{(\lambda+\gamma)\bar{e}_t}}{1 + \alpha\sigma^2 \frac{\lambda}{\lambda+\gamma}\left(e^{2(\lambda+\gamma)\bar{e}_t} - 1\right)}.$$

A GSP view: the optimal denoiser is a graph filter

- To build intuition, look at the denoising problem for a single signal at a fixed t. The noise model is fully known:

$$\mathbf{x}_t = \mathbf{H}_t \mathbf{x}_0 + \mathbf{e}, \qquad \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t), \qquad \mathbf{H}_t, \mathbf{\Sigma}_t \text{ known}.$$

- Many graph signals are smooth, so we add a Laplacian smoothing prior:

$$\hat{\mathbf{x}}_0 = g(\mathbf{x}_t) = \underset{\mathbf{x}_0}{\operatorname{argmin}} \ \|\mathbf{x}_t - \mathbf{H}_t \mathbf{x}_0\|^2_{\mathbf{\Sigma}_t^{-1}} + \alpha \|\mathbf{x}_0\|^2_{\mathbf{L}}.$$

Proposition: the solution is a graph filter

The minimizer is $\hat{\mathbf{x}}_0 = h(\mathbf{L})\mathbf{x}_t$, an ARMA graph filter.

$$h(\mathbf{L}) = \left(\mathbf{H}_t^\top \mathbf{\Sigma}_t^{-1} \mathbf{H}_t + \alpha \mathbf{L}\right)^{-1} \mathbf{H}_t^\top \mathbf{\Sigma}_t^{-1},$$

$$h(\lambda) = \frac{e^{(\lambda+\gamma)\bar{\epsilon}_t}}{1 + \alpha\sigma^2 \frac{\lambda}{\lambda+\gamma}\left(e^{2(\lambda+\gamma)\bar{\epsilon}_t} - 1\right)}.$$

- The backward process is therefore a sequence of graph-signal denoising problems – a classical, well-understood task in GSP.

Reading the filter: low-pass for noise, high-pass for detail

- $h(\lambda)$ is a tension between two terms:
  - $\Rightarrow$ an exponential numerator that replenishes high frequencies
  - $\Rightarrow$ a doubly-exponential denominator that suppresses them

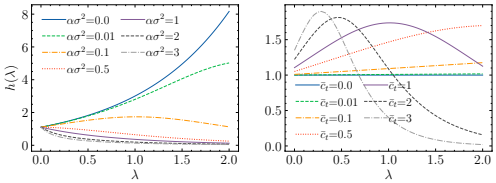


Figure. Frequency response $h(\lambda)$ for varying $\alpha\sigma^2$ (left) and $\bar{c}_t$ (right).

$$h(\lambda) = \frac{e^{(\lambda+\gamma)\bar{c}_t}}{1 + \alpha\sigma^2 \frac{\lambda}{\lambda+\gamma}(e^{2(\lambda+\gamma)\bar{c}_t} - 1)}$$

Reading the filter: low-pass for noise, high-pass for detail

- $h(\lambda)$ is a tension between two terms:
  - ⇒ an exponential numerator that replenishes high frequencies
  - ⇒ a doubly-exponential denominator that suppresses them

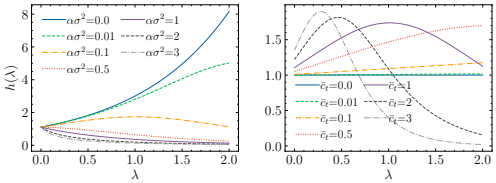


Figure. Frequency response $h(\lambda)$ for varying $\alpha\sigma^2$ (left) and $\bar{c}_t$ (right).

$$h(\lambda) = \frac{e^{(\lambda+\gamma)\bar{c}_t}}{1 + \alpha\sigma^2 \frac{\lambda}{\lambda+\gamma}(e^{2(\lambda+\gamma)\bar{c}_t} - 1)}$$

- **Noise level $\alpha\sigma^2$**:
  - ⇒ small ⇒ numerator wins ⇒ high-pass
  - ⇒ large ⇒ denominator wins ⇒ low-pass
- **Time $\bar{c}_t$**:
  - ⇒ large $\bar{c}_t$ ⇒ denominator dominates ⇒ low-pass
  - ⇒ small $\bar{c}_t$ ⇒ near-flat / high-pass

Reading the filter: low-pass for noise, high-pass for detail

- $h(\lambda)$ is a tension between two terms:
  - ⇒ an exponential numerator that replenishes high frequencies
  - ⇒ a doubly-exponential denominator that suppresses them

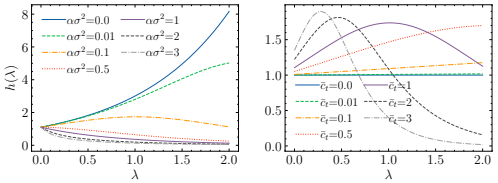


Figure. Frequency response $h(\lambda)$ for varying $\alpha\sigma^2$ (left) and $\bar{c}_t$ (right).

$$h(\lambda) = \frac{e^{(\lambda+\gamma)\bar{c}_t}}{1 + \alpha\sigma^2 \frac{\lambda}{\lambda+\gamma}(e^{2(\lambda+\gamma)\bar{c}_t} - 1)}$$

- **Noise level $\alpha\sigma^2$:**
  - ⇒ small ⇒ numerator wins ⇒ high-pass
  - ⇒ large ⇒ denominator wins ⇒ low-pass
- **Time $\bar{c}_t$:**
  - ⇒ large $\bar{c}_t$ ⇒ denominator dominates ⇒ low-pass
  - ⇒ small $\bar{c}_t$ ⇒ near-flat / high-pass

- **Takeaway:** generation unfolds coarse-to-fine: recover low-frequency structure first, then reintroduce high-frequency detail; this motivates a learnable GCNN that adapts across t.

Experiments: datasets, baselines, metrics

- **Datasets** (one fixed graph each):
  - ⇒ Synthetic SBM – two communities of 10 nodes, smooth signals
  - ⇒ METR-LA – traffic speeds at 207 sensors on the LA road network
  - ⇒ Molene – daily temperatures at 37 weather stations in France

- **Baselines:** variance-preserving (VPD) and variance-exploding (VED) diffusion
  - ⇒ both are graph-agnostic; under the SDE view they are directly comparable to GAD

- **Evaluation** (following graph-generation practice): three signal statistics

$$\mathrm{QV}(\mathbf{x}) = \mathbf{x}^\top \mathbf{L}\mathbf{x}, \qquad \mathrm{SC}(\mathbf{x}) = \frac{\sum_i \lambda_i |\tilde{x}_i|^2}{\sum_i |\tilde{x}_i|^2}, \qquad \mathrm{DC}(\mathbf{x}) = \mathrm{corr}(\mathbf{x}, \mathbf{d}),$$

smoothness (QV), spectrum (SC), degree alignment (DC) ⇒ MMD per metric ⇒ averaged into aMMD (lower is better).

- **Sampling:** Euler–Maruyama; each step is one GCNN pass, so we report quality vs. the number of steps

Results: GAD needs fewer sampling steps

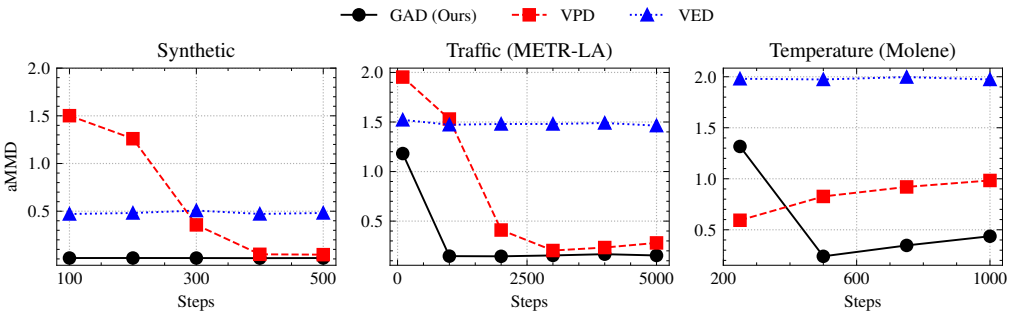


Figure. aMMD (lower is better) across sampling budgets.

Results: GAD needs fewer sampling steps

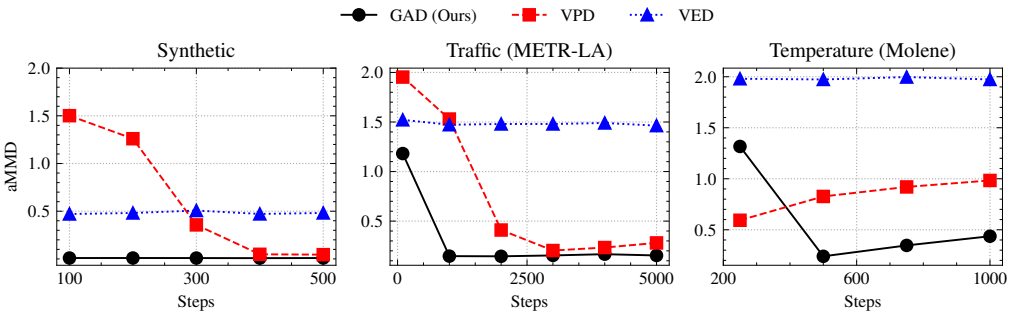


Figure. aMMD (lower is better) across sampling budgets.

- Main takeaway: GAD reaches low aMMD with far fewer reverse-SDE steps than graph-agnostic VPD/VED.

The gain is strongest in the low-step regime, where each saved step avoids one GCNN pass.

Conclusions

- **GAD:** a graph-aware diffusion model for generating signals on a known graph
- **Forward:** heat equation with the graph in the drift; closed-form GMRF marginals; converges to a graph-dependent stationary distribution; FCPS controls the noising rate
- **Backward:** via Tweedie, score estimation reduces to a sequence of graph-signal denoising problems; spectral analysis reveals a coarse-to-fine filter $\Rightarrow$ a learnable GCNN denoiser
- **Empirics:** consistent gains over graph-agnostic baselines, especially with few sampling steps

Conclusions

- **GAD:** a graph-aware diffusion model for generating signals on a known graph
- **Forward:** heat equation with the graph in the drift; closed-form GMRF marginals; converges to a graph-dependent stationary distribution; FCPS controls the noising rate
- **Backward:** via Tweedie, score estimation reduces to a sequence of graph-signal denoising problems; spectral analysis reveals a coarse-to-fine filter $\Rightarrow$ a learnable GCNN denoiser
- **Empirics:** consistent gains over graph-agnostic baselines, especially with few sampling steps
- **Where next:** multi-graph / transfer to larger graphs, per-eigenvalue schedulers, advanced SDE solvers

Thank you!

Questions?

- **Graph-Aware Diffusion for Signal Generation**
  - ⇒ code: `github.com/vimalkb7/gad`
  - ⇒ contact: `vkumarasamybal@tudelft.nl`

scan for the paper

Backup material

Backup: where is graph-signal generation useful?

- Sensor / monitoring networks: generate plausible temperature, traffic-speed, or air-quality fields over a fixed station graph – data augmentation, missing-value imputation, scenario simulation
- Probabilistic forecasting: sample multiple future graph signals to capture uncertainty in traffic, weather, energy demand, or sensor dynamics
- Recommender systems: synthesize user–item rating signals over a known user/item graph
- Wireless / power networks: simulate signals over a fixed infrastructure topology
- Privacy & stress-testing: produce realistic-yet-synthetic graph data when sharing the real signals is restricted
- The common thread: the graph is fixed and known, and we want new *signals* that respect it

Backup: the centering parameter γ

- $\mathbf{L}$ has a zero eigenvalue, so $-c_t\mathbf{L}$ does not converge. Adding $\gamma\mathbf{I}$ shifts the spectrum:

$$\mathbf{L}_\gamma = \mathbf{L} + \gamma\mathbf{I} \;\Rightarrow\; -c_t\mathbf{L}_\gamma \text{ is Hurwitz} \;\Rightarrow\; \text{convergence to } \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{L}_\gamma^{-1}).$$

Backup: the centering parameter γ

- $\mathbf{L}$ has a zero eigenvalue, so $-c_t\mathbf{L}$ does not converge. Adding $\gamma\mathbf{I}$ shifts the spectrum:

$$\mathbf{L}_\gamma = \mathbf{L} + \gamma\mathbf{I} \;\Rightarrow\; -c_t\mathbf{L}_\gamma \text{ is Hurwitz } \Rightarrow \text{ convergence to } \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{L}_\gamma^{-1}).$$

- A genuine trade-off in γ:
  - $\Rightarrow$ too small – low modes decay slowly; $\mathbf{L}_\gamma$ ill-conditioned; high-variance $\mathbf{\Sigma}_T$
  - $\Rightarrow$ too large – the geometry induced by the Laplacian is washed out

Backup: the centering parameter γ

- $\mathbf{L}$ has a zero eigenvalue, so $-c_t\mathbf{L}$ does not converge. Adding $\gamma\mathbf{I}$ shifts the spectrum:

$$\mathbf{L}_\gamma = \mathbf{L} + \gamma\mathbf{I} \;\Rightarrow\; -c_t\mathbf{L}_\gamma \text{ is Hurwitz} \;\Rightarrow\; \text{convergence to } \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{L}_\gamma^{-1}).$$

- A genuine trade-off in γ:
  - $\Rightarrow$ too small – low modes decay slowly; $\mathbf{L}_\gamma$ ill-conditioned; high-variance $\mathbf{\Sigma}_T$
  - $\Rightarrow$ too large – the geometry induced by the Laplacian is washed out

Remark (numerical stability)

We never form the matrix exponential directly; we use the eigendecomposition $\mathbf{H}_t = \mathbf{V}e^{-\bar{c}_t(\mathbf{\Lambda}+\gamma\mathbf{I})}\mathbf{V}^\top$.
This also hints at defining $\bar{c}_t$ per eigenvalue for independent mode control – a promising direction.

Backup: why a GCNN denoiser?

- The closed-form filter $h(\mathbf{L})$ is optimal only for a single signal at a single t

- In practice the denoiser must handle:
  - $\Rightarrow$ $\mathbf{x}_0$ in expectation over p_0 (the MMSE objective), not one signal
  - $\Rightarrow$ every noise level t, where the optimal filter sweeps low-pass $\rightarrow$ high-pass

Backup: why a GCNN denoiser?

- The closed-form filter $h(\mathbf{L})$ is optimal only for a single signal at a single t

- In practice the denoiser must handle:
  - $\Rightarrow$ $\mathbf{x}_0$ in expectation over p_0 (the MMSE objective), not one signal
  - $\Rightarrow$ every noise level t, where the optimal filter sweeps low-pass $\rightarrow$ high-pass

- A GCNN – a cascade of graph filters with nonlinearities – is the natural learnable generalization of $h(\mathbf{L})$
  - $\Rightarrow$ expressive enough to start from pure noise and adapt across t
  - $\Rightarrow$ permutation-equivariant, stable to perturbations, and transferable across graphs and timesteps

Backup: one reverse-diffusion step

- Each Euler–Maruyama step calls the GCNN once, converts its prediction into the Tweedie score, and takes one reverse-SDE step; iterating from $\mathbf{x}_T \sim p_\infty$ yields a sample $\mathbf{x}_0$