Second-Order Tensorial Partial Differential Equations on Graphs

Aref Einizade
LTCI, Télécom Paris
Institut Polytechnique de Paris
aref.einizade@telecom-paris.fr

Fragkiskos D. Malliaros
CentraleSupélec, Inria
Université Paris-Saclay
fragkiskos.malliaros@centralesupelec.fr

Jhony H. Giraldo
LTCI, Télécom Paris
Institut Polytechnique de Paris
jhony.giraldo@telecom-paris.fr

Abstract

Processing data that lies on multiple interacting (product) graphs is increasingly important in practical applications, yet existing methods are mostly restricted to discrete graph filtering. Tensorial partial differential equations on graphs (TPDEGs) offer a principled framework for modeling such multidomain data in a continuous setting. However, current continuous approaches are limited to first-order derivatives, which tend to dampen high-frequency signals and slow down information propagation. This makes these TPDEGs-based approaches less effective for capturing complex, multi-scale, and heterophilic structures. In this paper, we introduce second-order TPDEGs (So-TPDEGs) and propose the first theoretically grounded framework for second-order continuous product graph neural networks. Our approach leverages the separability of cosine kernels in Cartesian product graphs to implement efficient spectral decomposition, while naturally preserving high-frequency information. We provide rigorous theoretical analyses of stability under graph perturbations and over-smoothing behavior regarding spectral properties. Our theoretical results establish a robust foundation for advancing continuous graph learning across multiple practical domains.

1 Introduction

Tensors [1, 2], which generalize matrices to higher dimensions, arise in diverse domains such as hyperspectral imaging [3], video analytics [4], recommender systems [5], spatiotemporal modeling [6], and brain signal processing [7]. When tensors are linked to multiple interdependent graphs, they give rise to what we call *multidomain graph data* [8, 9, 10]. Unlike conventional graph machine learning, which typically considers a single graph [11, 9], this setting requires jointly processing across several interacting graph domains. Although such data structures are increasingly common in real-world applications, existing methods for learning from multidomain graph data remain scarce [12, 9, 13]. Therefore, developing graph learning techniques for tensorial representations holds great promise for advancing the theory and applications in graph machine learning.

Learning from multidomain graph data is challenging because it requires frameworks capable of capturing joint dependencies across several domain-specific graphs [12, 14]. A common strategy is to use product graphs [12, 15], a concept rooted in graph signal processing (GSP) [16, 17], where discrete filtering operations combine information across domains. However, these methods inherit well-known limitations of conventional graph neural networks (GNNs) [18, 19, 20, 21], such

NeurIPS 2025 Workshop: New Perspectives in Advancing Graph Machine Learning.

as over-smoothing and over-squashing, which restrict receptive fields and long-range modeling [22]. Discrete filtering-based models are also computationally expensive, relying on exhaustive hyperparameter searches, and are usually restricted to two-domain cases such as spatial–temporal graphs [12, 23, 15, 24].

Within standard non-tensorial GNNs, continuous GNNs (CGNNs) have emerged as an effective approach to mitigate the challenges of over-smoothing and over-squashing [25]. CGNNs formulate graph learning as solving partial differential equations (PDEs) on graphs—such as the heat or wave equations—through neural architectures that approximate these solutions [26, 25]. This leads to the exponential graph filter, a continuous, infinite-order generalization of discrete convolutional filters [25]. Because these filters are differentiable with respect to the receptive field, CGNNs can adaptively learn effective neighborhoods, enabling both local and global message passing [25, 22, 27]. While these frameworks have shown promise, they are mostly designed for single-graph settings. A notable exception is CITRUS [13], which extends continuous GNNs to multidomain data by introducing TPDEGs and leveraging product graph structures.

Despite its contributions, CITRUS is limited to first-order derivatives, which dampen high-frequency signals and slow information propagation—issues that are particularly problematic in oscillatory (periodic or high-frequency fine-grained) tasks such as spatiotemporal forecasting, heterophilic networks, and continuum mechanics simulations [28, 29, 30]. To overcome these limitations, we propose the first principled second-order TPDEGs (So-TPDEGs), which provide a stronger foundation for modeling multidomain data across multiple interacting graphs. Building on this formulation, we introduce a continuous graph learning model that efficiently solves So-TPDEGs using separable cosine oscillatory kernels on Cartesian product graphs. Our design avoids the computational burden of full spectral decompositions by reusing a small subset of eigenvalue decompositions from the factor graphs. We further provide rigorous theoretical analyses, showing that our model preserves stability under graph perturbations and controls the over-smoothing rate. The key contributions of this work are summarized below:

- We propose second-order tensorial PDEs on graphs (So-TPDEGs) as a unified framework for modeling multidomain graph data.
- We derive a continuous graph filtering mechanism over Cartesian product spaces, as a natural solution to So-TPDEGs.
- We provide comprehensive theoretical analyses of our model, establishing its stability and demonstrating its ability to alleviate over-smoothing.

2 Related Work

PDEs on graphs have recently attracted substantial attention, as reviewed in [25]. Several methods related to second-order PDEs—such as graph wave networks [28], hyperbolic-PDE GNNs [29], and graph-coupled oscillator networks [30]—have been proposed to capture oscillatory behaviors in graph signals. While effective, these models are designed for single-graph settings and do not provide a unified framework for multidomain data distributed over multiple interacting graphs.

Product graphs have been extensively studied in mathematics and signal processing [31, 32], but their research in GNNs remains limited. The graph-time convolutional neural network (GTCNN) [12, 23] and CITRUS [13] are, to the best of our knowledge, the only existing GNN-based frameworks explicitly designed for product graphs. GTCNN introduces a discrete filtering mechanism to jointly process spatial and temporal information, but suffers from three key limitations: (*i*) it requires costly grid searches to tune filter orders, (*ii*) polynomial filters restrict receptive fields and limit long-range modeling [22, 25], and (*iii*) it is inherently constrained to two-factor graphs (space and time). CITRUS [14] addresses some of these issues by introducing TPDEGs and providing a continuous filtering framework for multidomain data. However, it remains restricted to first-order derivatives, which dampen high-frequency components and slow down information propagation [28, 30].

In contrast to prior work, our approach introduces a second-order framework for multidomain learning, extending beyond the first-order formulation of CITRUS. By employing continuous separable cosine kernels on Cartesian product graphs, our model preserves high-frequency components and enables faster information propagation across domains. This design also allows receptive fields to be learned adaptively during training, eliminating the costly grid searches required by discrete

methods. Furthermore, by leveraging the spectral decomposition of factor graphs [22, 27], our method achieves controllable computational complexity. Importantly, the number of learnable parameters is independent of the number of factor graphs, ensuring scalability to large and complex multidomain settings.

3 Methodology

Preliminaries and notation. An undirected, weighted graph $\mathcal{G}$ with N vertices is represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of vertices and edges, respectively. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ characterizes the graph structure, with $\mathbf{A}_{ij} = \mathbf{A}_{ji} \geq 0$ indicating the connection strength between vertices i and j. We assume $\mathbf{A}_{ii} = 0$ for all i, meaning the graph contains no self-loops. The graph Laplacian $\mathbf{L} \in \mathbb{R}^{N\times N}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1})$ is the diagonal degree matrix and $\mathbf{1}$ is the all-ones vector. A *graph signal* is defined as a mapping $x : \mathcal{V} \to \mathbb{R}$, which assigns a real value to each node. Such a signal can be represented in vector form as $\mathbf{x} = [x_1, \ldots, x_N]^\top$. For a general vector $\mathbf{a} = [a_1, \ldots, a_N]^\top \in \mathbb{R}^{N\times 1}$, we define the element-wise cosine as $\cos(\mathbf{a}) = [\cos(a_1), \ldots, \cos(a_N)]^\top$. We denote the vectorization operator by $\mathrm{vec}(\cdot)$. Finally, using the Kronecker product $\otimes$, the Kronecker sum (also referred to as the Cartesian product) $\oplus$, and the Laplacian factors $\{\mathbf{L}_p \in \mathbb{R}^{N_p\times N_p}\}_{p=1}^P$, we introduce the following definitions.

$$\downarrow \oplus_{p=1}^P \mathbf{L}_p := \mathbf{L}_P \oplus \ldots \oplus \mathbf{L}_1, \;\; \otimes_{p=1}^P \mathbf{L}_p := \mathbf{L}_1 \otimes \ldots \otimes \mathbf{L}_P, \;\; \downarrow \otimes_{p=1}^P \mathbf{L}_p := \mathbf{L}_P \otimes \ldots \otimes \mathbf{L}_1, \quad (1)$$

where the Laplacian matrix of the Cartesian product between two factor graphs can be stated as [31]:

$$\oplus_{p=1}^2 \mathbf{L}_p = \mathbf{L}_1 \oplus \mathbf{L}_2 = \mathbf{L}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{L}_2. \quad (2)$$

Here, $\mathbf{I}_n$ denotes the identity matrix of size n. An illustration of a Cartesian graph product between three factor graphs is shown in Figure 1(a). A relationship analogous to (2) also applies to adjacency matrices [32, 33]. We define a D-dimensional tensor as $\underline{\mathbf{U}} \in \mathbb{R}^{N_1\times\cdots\times N_D}$. The matrix $\underline{\mathbf{U}}_{(i)} \in \mathbb{R}^{N_i\times\left[\prod_{r=1,r\neq i}^D N_r\right]}$ represents the *i-th mode matricization* (unfolding) of the tensor $\underline{\mathbf{U}}$ [1]. The *mode-i tensor multiplication* of a tensor $\underline{\mathbf{U}}$ by a matrix $\mathbf{X} \in \mathbb{R}^{m\times N_i}$ is defined as $\underline{\mathbf{G}} = \underline{\mathbf{U}} \times_i \mathbf{X}$, where $\underline{\mathbf{G}} \in \mathbb{R}^{N_1\times\cdots\times N_{i-1}\times m\times N_{i+1}\times\cdots\times N_D}$ [1]. All the proofs of theorems and propositions of this paper are provided in the appendices.

3.1 Second-Order Continuous Product Graph Neural Networks

We first propose the second-order generative PDE for product graphs as follows:

Definition 3.1 (Second-order Tensorial PDE on graphs (So-TPDEG))**.** Let $\tilde{\underline{\mathbf{U}}}_t \in \mathbb{R}^{N_1\times N_2\times\cdots\times N_P}$ denote a multidomain tensor whose elements vary as functions of time t. We further define the So-TPDEG with $\{\mathbf{L}_p \in \mathbb{R}^{N_p\times N_p}\}_{p=1}^P$ as the set of domain-specific Laplacian matrices corresponding to each factor graph, as follows:

$$\frac{\partial^2 \underline{\mathbf{U}}_t}{\partial\, t^2} = -\sum_{i=1}^P \underline{\mathbf{U}}_t \times_i \mathbf{L}_i^2 - 2 \sum_{1\leq i<j\leq P} \underline{\mathbf{U}}_t \times_i \mathbf{L}_i \times_j \mathbf{L}_j. \quad (3)$$

Here, there are two important key differences with the first-order TPDEGs [13]: (*i*) The first term in the RHS applies $\mathbf{L}_i^2$, compared to TPDEG, which exploits $\mathbf{L}_i$. Therefore, So-TPDEG potentially searches for further neighborhoods. (*ii*) The second term in the RHS of (3) models the cross-interaction between factor graphs, compared to the TPDEG relying only on the independencies of the domain-specific factor graphs.

Theorem 3.2. *Assuming $\tilde{\underline{\mathbf{U}}}_0$ as the initial value of $\tilde{\underline{\mathbf{U}}}_t$, the solution to the So-TPDEG in* (3) *(with the assumption of $\left.\frac{\partial \mathbf{U}_t}{\partial t}\right|_{t=0} = 0$) is given by:*

$$\tilde{\underline{\mathbf{U}}}_t = \sum_{\substack{\boldsymbol{\epsilon}\in\{0,1\}^P \\ |\boldsymbol{\epsilon}| \text{ even}}} (-1)^{|\boldsymbol{\epsilon}|/2}\, \tilde{\underline{\mathbf{U}}}_0 \times_1 g_t^{(\epsilon_1)}(\mathbf{L}_1) \times_2 g_t^{(\epsilon_2)}(\mathbf{L}_2) \cdots \times_P g_t^{(\epsilon_P)}(\mathbf{L}_P), \quad (4)$$

where $g_t^{(0)}(\mathbf{L}_i) = \cos(t\mathbf{L}_i)$, $g_t^{(1)}(\mathbf{L}_i) = \sin(t\mathbf{L}_i)$, $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_P) \in \{0,1\}^P$, with $\{0,1\}^P$ being the Cartesian product between P sets of $\{0,1\}$. Besides, $|\boldsymbol{\epsilon}| := \sum_{i=1}^P \epsilon_i$.

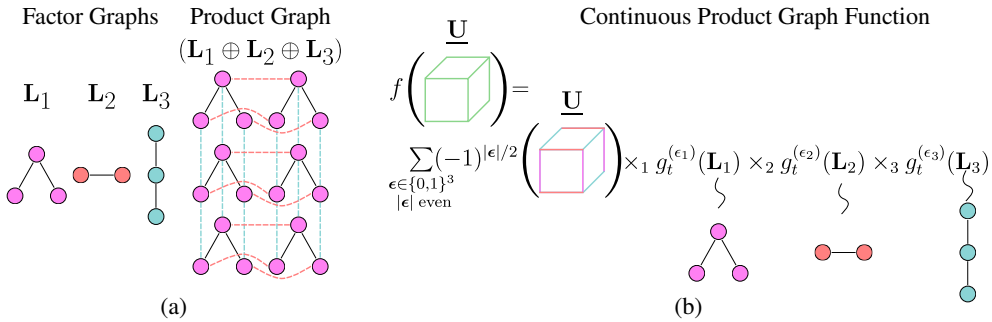

Figure 1: Illustration of key concepts of our model. (a) Cartesian product between three-factor graphs. (b) Continous product graph function operating on the multidomain graph data $\underline{\mathbf{U}}$.

Let $\underline{\mathbf{U}}_l \in \mathbb{R}^{N_1 \times \ldots \times N_P \times F_l}$ be the input tensor, $\mathbf{W}_l \in \mathbb{R}^{F_l \times F_{l+1}}$ a matrix of learnable parameters, t_l the learnable graph receptive field, and $f(\underline{\mathbf{U}}_l) \in \mathbb{R}^{N_1 \times \ldots \times N_P \times F_{l+1}}$ the output tensor. Using Theorem 3.2, we can define the core function of our framework as follows:

$$f(\underline{\mathbf{U}}_l) = \sum_{\substack{\boldsymbol{\epsilon} \in \{0,1\}^P \\ |\boldsymbol{\epsilon}| \text{ even}}} (-1)^{|\boldsymbol{\epsilon}|/2}\, \underline{\mathbf{U}}_l \times_1 g_{t_l}^{(\epsilon_1)}(\mathbf{L}_1) \times_2 g_{t_l}^{(\epsilon_2)}(\mathbf{L}_2) \cdots \times_P g_{t_l}^{(\epsilon_P)}(\mathbf{L}_P) \times_{P+1} \mathbf{W}_l^\top. \tag{5}$$

Therefore, the core graph filtering operation is obtained on a product graph as follows (see Figure 1(b) for a high-level description):

Proposition 3.3. *The core function of our framework in* (5) *can be rewritten as:*

$$\left[f(\underline{\mathbf{U}}_l)_{(P+1)}\right]^\top = \cos(t_l \mathbf{L}_\diamond)[\underline{\mathbf{U}}_{l(P+1)}]^\top \mathbf{W}_l, \tag{6}$$

where $\mathbf{L}_\diamond := \downarrow \oplus_{p=1}^P \mathbf{L}_p$ *is the Laplacian of the Cartesian product graph.*

Proposition 3.3 provides the theoretical basis for the practical implementation of our framework. To compute the solution efficiently, we can use the spectral decompositions of the factor and product Laplacians $\{\mathbf{L}_p = \mathbf{V}_p \boldsymbol{\Lambda}_p \mathbf{V}_p^\top\}_{p=1}^P$ and $\{\mathbf{L}_\diamond = \mathbf{V}_\diamond \boldsymbol{\Lambda}_\diamond \mathbf{V}_\diamond^\top\}$, respectively. Here, the eigenvectors of the product graph are obtained as $\mathbf{V}_\diamond = \downarrow \otimes_{p=1}^P \mathbf{V}_p$, and the corresponding eigenvalues follow $\boldsymbol{\Lambda}_\diamond = \downarrow \oplus_{p=1}^P \boldsymbol{\Lambda}_p$ [31]. For each factor graph, we choose the top $K_p \leq N_p$ eigenvalue–eigenvector pairs to reduce computational cost. More precisely, it can be shown that [27, 22] Equation (6) can be equivalently expressed in the following form:

$$\left[f(\underline{\mathbf{U}}_l)_{(P+1)}\right]^\top = \mathbf{V}_\diamond^{(K_p)} \left(\underbrace{\overbrace{[\tilde{\boldsymbol{\lambda}}_l | \ldots | \tilde{\boldsymbol{\lambda}}_l]}^{F_l \text{ times}}}_{\tilde{\boldsymbol{\Lambda}}_l} \odot \left(\mathbf{V}_\diamond^{(K_p)^\top} \left[\underline{\mathbf{U}}_{l(P+1)}\right]^\top \right) \right) \mathbf{W}_l, \tag{7}$$

$$\text{with} \qquad \tilde{\boldsymbol{\lambda}}_l = \sum_{\substack{\boldsymbol{\epsilon} \in \{0,1\}^P \\ |\boldsymbol{\epsilon}| \text{ even}}} (-1)^{|\boldsymbol{\epsilon}|/2} \downarrow \otimes_{p=1}^P g_{t_l}^{(\epsilon_p)}(\boldsymbol{\lambda}_p^{(K_p)}), \qquad \mathbf{V}_\diamond^{(K_p)} = \downarrow \otimes_{p=1}^P \mathbf{V}_p^{(K_p)}. \tag{8}$$

Here, $\boldsymbol{\lambda}_p^{(K_p)} \in \mathbb{R}^{K_p \times 1}$ and $\mathbf{V}_p^{(K_p)} \in \mathbb{R}^{N_p \times K_p}$ represent the top $K_p \leq N_p$ eigenvalues and their corresponding eigenvectors of $\mathbf{L}_p$, selected according to the largest magnitudes of the eigenvalues. The symbol $\odot$ denotes the element-wise (Hadamard) product. Using these components, the output of the (l)-th layer in our model is expressed as $\underline{\mathbf{U}}_{l+1(P+1)} = \sigma\left(f(\underline{\mathbf{U}}_l)_{(P+1)}\right)$, where $\sigma(\cdot)$ denotes a suitable nonlinear activation function.

Remark 3.4. To enhance the flexibility of our framework, we introduce factor-specific learnable graph receptive fields, denoted by $\{t_l^{(p)}\}_{p=1}^P$. Using these parameters, $\tilde{\boldsymbol{\lambda}}_l$ in (8) can be reformulated as $\tilde{\boldsymbol{\lambda}}_l = \sum_{\boldsymbol{\epsilon} \in \{0,1\}^P,\, |\boldsymbol{\epsilon}| \text{ even}} (-1)^{|\boldsymbol{\epsilon}|/2} \downarrow \otimes_{p=1}^P g_{t_l^{(p)}}^{(\epsilon_p)}(\boldsymbol{\lambda}_p^{(K_p)})$, allowing each factor graph to learn its own adaptive receptive field. This idea can be further extended to channel-wise receptive fields $\{t_l^{(p,c)}\}_{p=1,c=1}^{P,F_l}$, where the c-th column of $\tilde{\boldsymbol{\Lambda}}_l$ in (7) is computed as $\tilde{\boldsymbol{\lambda}}_{k,c} = \sum_{\boldsymbol{\epsilon} \in \{0,1\}^P,\, |\boldsymbol{\epsilon}| \text{ even}} (-1)^{|\boldsymbol{\epsilon}|/2} \downarrow \otimes_{p=1}^P g_{t_l^{(p,c)}}^{(\epsilon_p)}(\boldsymbol{\lambda}_p^{(K_p)})$.

Remark 3.5. The eigenvalue decomposition (EVD) of the product graph Laplacian $\mathbf{L}_\diamond$ in (6) has a computational cost of $\mathcal{O}\left(\left[\prod_{p=1}^{P} N_p\right]^3\right)$ in the general case. However, leveraging the separable structure of product graphs significantly reduces the complexity to $\mathcal{O}\left(\sum_{p=1}^{P} N_p^3\right)$ in (7), since EVD is performed independently on each factor graph. Furthermore, if we retain only the top K_p most significant eigenvalue-eigenvector pairs [22] per factor graph, the cost of spectral decomposition is further reduced to $\mathcal{O}(N_p^2 K_p)$ for each factor, resulting in an overall complexity of $\mathcal{O}\left(\sum_{p=1}^{P} N_p^2 K_p\right)$.

3.2 Stability Analysis

We investigate the stability of So-CITRUS under potential perturbations in the factor graphs. Following the approach in [34], a perturbation on the p-th graph is modeled by adding an error matrix $\mathbf{E}_p$ to its Laplacian $\mathbf{L}_p$, where the magnitude of the perturbation is bounded by ε_p with respect to a matrix norm $|||\cdot|||$, as follows:

$$\tilde{\mathbf{L}}_p = \mathbf{L}_p + \mathbf{E}_p; \quad |||\mathbf{E}_p||| \leq \varepsilon_p. \tag{9}$$

Proposition 3.6. *Let $\mathbf{L}_\diamond$ and $\tilde{\mathbf{L}}_\diamond$ denote the original and perturbed Laplacian matrices of the Cartesian product graph, where the perturbation for each factor graph follows the model described in* (9)*. Under this formulation, the perturbation matrix $\mathbf{E}$ admits the Cartesian structure, such that $\mathbf{E} = \oplus_{p=1}^{P} \mathbf{E}_p$, and the following relationship holds:*

$$\tilde{\mathbf{L}}_\diamond = \mathbf{L}_\diamond + \mathbf{E}; \quad |||\mathbf{E}||| \leq \sum_{p=1}^{P} \varepsilon_p, \tag{10}$$

Let $\varphi(u,t)$ and $\tilde{\varphi}(u,t)$ denote the outputs of our model under the original and perturbed Laplacian $\mathbf{L}_\diamond$, respectively. The Laplacian $\mathbf{L}_\diamond$ governs the evolution of $\underline{\mathbf{U}}_{t(P+1)}$ through the flow PDE $\frac{\partial^2 \varphi(u,t)}{\partial t^2} = -\mathbf{L}_\diamond^2 \varphi(u,t)$ as described in (6), with the corresponding solution $\varphi(u,t) = \cos(t\mathbf{L}_\diamond)\varphi(u,0)$, assuming $\left.\frac{\partial \varphi(u,t)}{\partial t}\right|_{t=0} = 0$. The following theorem then establishes that, for Cartesian product graphs, the error bound on the stability of our model admits the integrated sum of stability over the factor graphs.

Theorem 3.7. *Consider a Cartesian product graph (assumed to have no isolated nodes) with Laplacians $\mathbf{L}_\diamond$ and $\tilde{\mathbf{L}}_\diamond$ as defined in* (6)*. Suppose the perturbation on each factor graph is modeled as $\tilde{\mathbf{L}}_p = \mathbf{L}_p + \mathbf{E}_p$, with factor-wise error bounds $\{|||\mathbf{E}_p||| \leq \varepsilon_p\}_{p=1}^{P}$ as described in Proposition 3.6. Under this setup, the stability of our model can be characterized by comparing the outputs $\varphi(u,t)$ and $\tilde{\varphi}(u,t)$ corresponding to the true and perturbed Laplacians, respectively. Particularly, the overall stability bound is given by the sum of the individual stability from each factor graph:*

$$\|\varphi(u,t) - \tilde{\varphi}(u,t)\| \leq t\, |||\varphi(u,0)||| \left(\sum_{p=1}^{P} \varepsilon_p\right) = \sum_{p=1}^{P} \mathcal{O}(\varepsilon_p). \tag{11}$$

Theorem 3.7 establishes that the solution of the So-TPDEG (4) remains stable with respect to the magnitude of perturbations in the factor graphs. This robustness is particularly important when the factor graphs are subject to errors.

3.3 Over-smoothing Analysis

Over-smoothing in GNNs is commonly characterized by the decay of Dirichlet energy as the number of layers increases. This phenomenon has been theoretically studied in prior works [35, 36, 25] and can be formulated for the output of a continuous GNN [25]. Specifically, consider the GNN output $\mathbf{U}_t = [\mathbf{u}(t)_1, \ldots, \mathbf{u}(t)_N]^\top$, where $\mathbf{u}(t)_i$ represents the feature vector of node i, which has degree deg_i. Inspired by the previous studies [13], the Dirichlet energy can then be defined as:

$$\lim_{t\to\infty} E(\mathbf{U}_t) \to 0, \quad \text{where} \quad E(\mathbf{U}) := \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \left\| \frac{\mathbf{u}_i}{\sqrt{\text{deg}_i}} - \frac{\mathbf{u}_j}{\sqrt{\text{deg}_j}} \right\|^2 = \text{tr}(\mathbf{U}^\top \hat{\mathbf{L}} \mathbf{U}). \tag{12}$$

Motivated by (12), we generalize the concept of Dirichlet energy to tensors as follows:

Definition 3.8. (Tensorial Dirichlet Energy [13]). Let $\{\hat{\mathbf{L}}_p\}_{p=1}^{P}$ denote the normalized factor Laplacians. For a tensor $\underline{\mathbf{U}} \in \mathbb{R}^{N_1 \times \cdots \times N_P \times F}$, the Tensorial Dirichlet Energy is defined by:

$$E(\underline{\mathbf{U}}) := \frac{1}{P} \sum_{f=1}^{F} \sum_{p=1}^{P} \mathrm{tr}\Big(\tilde{\mathbf{U}}_{f_{(p)}}^{\top} \hat{\mathbf{L}}_p \tilde{\mathbf{U}}_{f_{(p)}} \Big), \tag{13}$$

where $\tilde{\mathbf{U}}_{f_{(p)}} = \underline{\mathbf{U}}[:, \ldots, :, f]_{(p)} \in \mathbb{R}^{N_p \times \prod_{i \neq p}^{P} N_i}$ represents the p-th mode matricization of the f-th slice of $\underline{\mathbf{U}}$ along the $(P+1)$-th dimension.

Drawing on prior work in the GSP literature [33, 37, 38], the tensorial Dirichlet energy $E(\underline{\mathbf{U}})$ defined in (13) can alternatively be expressed in matrix form as $E(\underline{\mathbf{U}}) = \mathrm{tr}\Big(\tilde{\mathbf{U}}^{\top} \hat{\mathbf{L}} \tilde{\mathbf{U}}\Big)$, where $\tilde{\mathbf{U}} \in \mathbb{R}^{[\prod_{p=1}^{P} N_p] \times F}$, with columns given by $\tilde{\mathbf{U}}[:, f] = \mathrm{vec}(\underline{\mathbf{U}}[:, \ldots, :, f])$, and where the normalized product graph Laplacian is defined as $\hat{\mathbf{L}} := \frac{1}{P} \oplus_{p=1}^{P} \hat{\mathbf{L}}_p = \oplus_{p=1}^{P} \frac{\hat{\mathbf{L}}_p}{P}$, implying a useful spectral property that the spectrum of $\hat{\mathbf{L}}$ spans the interval of $[0, 2]$ [13]. From (13), we note that the separability of Cartesian heat kernels naturally extends to analyzing over-smoothing across the different modes of the data. This property is particularly advantageous when the factor graphs exhibit diverse characteristics. We now proceed with a formal analysis of over-smoothing in our model. Specifically, for the l-th layer, which consists of H_l hidden MLP layers, nonlinear activation functions $\sigma(\cdot)$, and learnable weight matrices $\{\mathbf{W}_{lh}\}_{h=1}^{H_l}$, we define:

$$\mathbf{X}_{l+1} := f_l(\mathbf{X}_l),\ f_l(\mathbf{X}) := \mathrm{MLP}_l(\cos(\hat{\mathbf{L}}^{(t)})\mathbf{X}),\ \mathrm{MLP}_l(\mathbf{X}) := \sigma(\ldots \sigma(\sigma(\mathbf{X})\mathbf{W}_{l1})\mathbf{W}_{l2} \ldots \mathbf{W}_{lH_l}), \tag{14}$$

Here, $\mathbf{X}_0$ denotes the initial node feature matrix, and $f_l(\cdot)$ represents a generalized form of (6). The term $cos(\hat{\mathbf{L}}^{(t)})$ is defined as $\hat{\mathbf{L}} := \frac{1}{P} \oplus_{p=1}^{P} \hat{\mathbf{L}}_p$, where $t^{(p)}$ corresponds to the receptive field of the p-th factor graph. The following theorem then provides the formal criteria for over-smoothing.

Theorem 3.9. *For the product Laplacian $\hat{\mathbf{L}} := \frac{1}{P} \oplus_{p=1}^{P} \hat{\mathbf{L}}_p$ with domain-specific receptive field $t^{(i)}$ for the i-th factor graph and the activations $\sigma(\cdot)$ in* (14) *being ReLU or Leaky ReLU, we have:*

$$E(\mathbf{X}^{(l)}) \leq (s \cos^2(t\lambda_{\phi}^{\diamond}))^l E(\mathbf{X}),$$
$$\textit{where}\ \cos(t\,\lambda_{\phi}^{\diamond}) = \sum_{\substack{S \subseteq \{1,\ldots,P\} \\ |S| \text{ even}}} (-1)^{|S|/2} \prod_{p \in S} \sin(t\,\lambda_{i_p}^{(p)}) \prod_{p \notin S} \cos(t\,\lambda_{i_p}^{(p)}). \tag{15}$$

Here, $s := \sup_{l \in \mathbb{N}_+} s_l$ and $s_l := \prod_{h=1}^{H_l} s_{lh}$ with s_{lh} being the square of maximum singular value of $\mathbf{W}_{lh}^{\top}$. Additionally, $\lambda_{i_p}^{(p)}$ is the i_p-th eigenvalue of the p-th normalized factor graph such that $\cos^2(t\lambda_{\phi}^{\diamond}) = \max_i \cos^2(t\lambda_i^{\diamond})$, where $\lambda_i^{\diamond}$ is the i-th non-zero eigenvalue of $\hat{\mathbf{L}}_{\diamond}$ and $\lambda_{\phi}^{\diamond} = \sum_{p=1}^{P} \lambda_{i_p}^{(p)}$. Besides, $\cos^2(t\lambda_{\phi}^{\diamond}) = \max_i \cos^2(t\lambda_i^{\diamond})$, where $\lambda_i^{\diamond}$ is the i-th non-zero eigenvalue of $\hat{\mathbf{L}}_{\diamond}$.

Corollary 3.10. *If $s \cos^2(t\lambda_{\phi}^{\diamond}) < 1$ and $l \to \infty$, $E(\mathbf{X}_l)$ exponentially converges to 0, leading to:*

$$t \in \bigcup_{k \in \mathbb{Z}} \frac{1}{2\lambda_{\phi}^{\diamond}} \left(2k\pi + \arccos\left(\frac{2}{s} - 1\right),\ 2(k+1)\pi - \arccos\left(\frac{2}{s} - 1\right) \right). \tag{16}$$

Theorem 3.9 and Corollary 3.10 demonstrate that the oversmoothing rate is influenced by three key factors: the learnable weight matrices (through s), the spectral characteristics of the factor and product graph (via $\lambda_{\phi}^{\diamond}$), and the graph receptive field t. Unlike first-order TPDEGs [13], which yield only a single lower bound on t and consequently suppress high-frequency components, Eqn. (16) produces infinitely many valid intervals. This flexibility enables control over oversmoothing by tuning t and naturally leads to oscillatory solutions that act as an inherent band-pass filter on the factor graphs.

4 Conclusion and Discussion

In this work, we introduced So-TPDEG, a second-order tensorial PDE framework designed to process data over multiple interacting (product) graphs, supported by a rigorous theoretical foundation.

Leveraging this formulation, we develop a second-order continuous product GNN, a novel model that effectively solves these PDEs while providing a principled treatment of stability and over-smoothing. Compared to current first-order TPDEG models, So-TPDEG does not dampen the medium to high-frequency components (because it is not a low-pass graph filter [16]), which might be applicable in some real-world applications with periodic patterns or fine-grained details [28, 29, 30].

In future work, we will complement these theoretical advances with systematic experimental validation on real-world benchmarks, as well as experimental validation of the theoretical statements describing its foundational aspects. A natural direction is spatiotemporal forecasting, where interactions between spatial and temporal graphs across multiple time steps, together with seasonal or yearly periodicities, provide a compelling testbed for our model.

5 Acknowledgment

This research was supported by the ANR projects DeSNAP (ANR-24-CE23-1895-01) and GraphIA (ANR-20-CE23-0009-01).

References

[1] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[3] X. Song, K. Wu, and L. Chai, "Brain network analysis of schizophrenia patients based on hypergraph signal processing," *IEEE Transactions on Image Processing*, vol. 32, pp. 4964–4976, 2023.

[4] N. Shahid, F. Grassi, and P. Vandergheynst, "Tensor robust PCA on graphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[5] G. Ortiz-Jiménez, M. Coutino, S. P. Chepuri, and G. Leus, "Sparse sampling for inverse problems with tensors," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3272–3286, 2019.

[6] A. Cini, I. Marisca, D. Zambon, and C. Alippi, "Graph deep learning for time series forecasting," *ACM Computing Surveys*, vol. 57, no. 12, pp. 1–34, 2025.

[7] G. Mishne, R. Talmon, R. Meir, J. Schiller, M. Lavzin, U. Dubin, and R. R. Coifman, "Hierarchical coupled-geometry analysis for neuronal structure and activity pattern discovery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 7, pp. 1238–1253, 2016.

[8] Y. L. Xu, K. Konstantinidis, and D. P. Mandic, "Multi-graph tensor networks," in *Advances in Neural Information Processing Systems - Workshop*, 2020.

[9] Y. L. Xu, K. Konstantinidis, and D. P. Mandic, "Graph tensor networks: An intuitive framework for designing large-scale neural learning systems on multiple domains," *arXiv preprint arXiv:2303.13565*, 2023.

[10] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Advances in Neural Information Processing Systems*, 2017.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[12] M. Sabbaqi and E. Isufi, "Graph-time convolutional neural networks: Architecture and theoretical analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 14625–14638, 2023.

[13] A. Einizade, F. D. Malliaros, and J. H. Giraldo, "Continuous product graph neural networks," in *Advances in Neural Information Processing Systems*, 2024.

[14] A. Einizade and S. H. Sardouie, "Learning product graphs from spectral templates," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 9, pp. 357–372, 2023.

[15] A. Einizade, S. Nasiri, S. H. Sardouie, and G. D. Clifford, "ProductGraphSleepNet: Sleep staging using product spatio-temporal graph learning with attentive temporal aggregation," *Neural Networks*, vol. 164, pp. 667–680, 2023.

[16] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[17] G. Leus, A. G. Marques, J. M. Moura, A. Ortega, and D. I. Shuman, "Graph signal processing: History, development, impact, and outlook," *IEEE Signal Processing Magazine*, vol. 40, no. 4, pp. 49–60, 2023.

[18] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[19] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *International Conference on Learning Representations*, 2020.

[20] C. Cai and Y. Wang, "A note on over-smoothing for graph neural networks," in *International Conference on Machine Learning - Workshop*, 2020.

[21] J. H. Giraldo, K. Skianis, T. Bouwmans, and F. D. Malliaros, "On the trade-off between over-smoothing and over-squashing in deep graph neural networks," in *ACM International Conference on Information and Knowledge Management*, 2023.

[22] M. Behmanesh, M. Krahn, and M. Ovsjanikov, "TIDE: Time derivative diffusion for deep learning on graphs," in *International Conference on Machine Learning*, 2023.

[23] E. Isufi and G. Mazzola, "Graph-time convolutional neural networks," in *IEEE Data Science and Learning Workshop*, 2021.

[24] J. A. Castro-Correa, J. H. Giraldo, M. Badiey, and F. D. Malliaros, "Gegenbauer graph neural networks for time-varying signal reconstruction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 9, pp. 11734–11745, 2024.

[25] A. Han, D. Shi, L. Lin, and J. Gao, "From continuous dynamics to graph neural networks: Neural diffusion and beyond," *Transactions on Machine Learning Research*, 2024.

[26] L.-P. Xhonneux, M. Qu, and J. Tang, "Continuous graph neural networks," in *International Conference on Machine Learning*, 2020.

[27] N. Sharp, S. Attaiki, K. Crane, and M. Ovsjanikov, "DiffusionNet: Discretization agnostic learning on surfaces," *ACM Transactions on Graphics*, vol. 41, no. 3, pp. 1–16, 2022.

[28] J. Yue, H. Li, J. Sheng, Y. Guo, X. Zhang, C. Zhou, T. Liu, and L. Guo, "Graph wave networks," in *Proceedings of the ACM on Web Conference*, 2025.

[29] J. Yue, H. Li, J. Sheng, X. Li, T. Su, T. Liu, and L. Guo, "Hyperbolic-PDE GNN: Spectral graph neural networks in the perspective of a system of hyperbolic partial differential equations," in *International Conference on Machine Learning*, 2025.

[30] T. K. Rusch, B. Chamberlain, J. Rowbottom, S. Mishra, and M. Bronstein, "Graph-coupled oscillator networks," in *International Conference on Machine Learning*, 2022.

[31] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[32] R. Hammack, W. Imrich, and S. Klavžar, *Handbook of product graphs*. CRC press, 2011.

[33] S. K. Kadambari and S. P. Chepuri, "Product graph learning from multi-domain data with sparsity and rank constraints," *IEEE Transactions on Signal Processing*, vol. 69, pp. 5665–5680, 2021.

[34] Y. Song, Q. Kang, S. Wang, K. Zhao, and W. P. Tay, "On the robustness of graph neural diffusion to topology perturbations," in *Advances in Neural Information Processing Systems*, 2022.

[35] F. D. Giovanni, J. Rowbottom, B. P. Chamberlain, T. Markovich, and M. M. Bronstein, "Understanding convolution on graphs via energies," *Transactions on Machine Learning Research*, 2023.

[36] T. K. Rusch, M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," *SAM Research Report*, 2023.

[37] M. Indibi and S. Aviyente, "Low-rank and smooth tensor recovery on Cartesian product graphs," in *IEEE International Conference on Sampling Theory and Applications*, 2023.

[38] J. S. Stanley, E. C. Chi, and G. Mishne, "Multiway graph signal processing on tensors: Integrative analysis of irregular geometries," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 160–173, 2020.

A Proof Theorem 3.2

First, we state and prove the following Lemma:

Lemma A.1. *By defining $g_t^{(0)}(\mathbf{L}_p) := \cos(t\mathbf{L}_p)$ and $g_t^{(1)}(\mathbf{L}_p) := \sin(t\mathbf{L}_p)$ for a specific p, the cosine filter applied in the product graph can be stated in terms of the factor graphs as follows:*

$$\cos(t\mathbf{L}_\diamond) = \sum_{\substack{\boldsymbol{\epsilon}\in\{0,1\}^P \\ |\boldsymbol{\epsilon}| \text{ even}}} (-1)^{|\boldsymbol{\epsilon}|/2} \downarrow \otimes_{p=1}^{P} g_t^{(\epsilon_p)}(\mathbf{L}_p), \tag{17}$$

Proof. Let $\mathbf{L}_\diamond = \downarrow \oplus_{p=1}^{P} \mathbf{L}_p$. Using the Kronecker-sum property of the matrix exponential,

$$e^{\mathrm{i}t\mathbf{L}_\diamond} = \downarrow \otimes_{p=1}^{P} e^{\mathrm{i}t\mathbf{L}_p}. \tag{18}$$

Since $\cos(t\mathbf{L}_\diamond) = \Re\big(e^{\mathrm{i}t\mathbf{L}_\diamond}\big)$, where $\Re(.)$ computes the real part of a complex number, it suffices to expand the right-hand side. For each p, one can write:

$$e^{\mathrm{i}t\mathbf{L}_i} = \cos(t\mathbf{L}_i) + \mathrm{i}\sin(t\mathbf{L}_i) = g_t^{(0)}(\mathbf{L}_p) + \mathrm{i}\, g_t^{(1)}(\mathbf{L}_p), \tag{19}$$

with $g_t^{(0)}(\mathbf{L}_p) := \cos(t\mathbf{L}_p)$ and $g_t^{(1)}(\mathbf{L}_p) := \sin(t\mathbf{L}_p)$. The Kronecker product of these sums expands as:

$$\downarrow \otimes_{p=1}^{P} \big(g_t^{(0)}(\mathbf{L}_p) + \mathrm{i}\, g_t^{(1)}(\mathbf{L}_p)\big) = \sum_{\boldsymbol{\epsilon}\in\{0,1\}^P} \mathrm{i}^{|\boldsymbol{\epsilon}|} \downarrow \otimes_{p=1}^{P} g_t^{(\epsilon_p)}((\mathbf{L}_p)), \tag{20}$$

because choosing $\epsilon_p = 1$ picks the $\mathrm{i}\, g_t^{(1)}(\mathbf{L}_p)$ term in factor p, contributing one factor of p per chosen sine.

Taking real parts, only multi-indices with even $|\boldsymbol{\epsilon}| = \sum_p \epsilon_p$ remain, and for $|\boldsymbol{\epsilon}| = 2k$ we have $\Re(\mathrm{i}^{2k}) = \Re((-1)^k) = (-1)^k$. Hence:

$$\cos(t\mathbf{L}_\diamond) = \sum_{\substack{\boldsymbol{\epsilon}\in\{0,1\}^P \\ |\boldsymbol{\epsilon}| \text{ even}}} (-1)^{|\boldsymbol{\epsilon}|/2} \downarrow \otimes_{p=1}^{P} g_t^{(\epsilon_p)}(\mathbf{L}_p), \tag{21}$$

which is the claimed formula. □

Now, using Lemma A.1 and by performing vectorization on two sides of the equation (4):

$$\mathrm{vec}(\underline{\mathbf{U}}_t) = \sum_{\substack{\boldsymbol{\epsilon}\in\{0,1\}^P \\ |\boldsymbol{\epsilon}| \text{ even}}} (-1)^{|\boldsymbol{\epsilon}|/2} \downarrow \otimes_{p=1}^{P} g_t^{(\epsilon_p)}(\mathbf{L}_p)\mathrm{vec}(\underline{\mathbf{U}}_0) = \cos(t\mathbf{L}_\diamond)\mathrm{vec}(\underline{\mathbf{U}}_0), \tag{22}$$

By defining $\mathbf{u}_t := \mathrm{vec}(\underline{\mathbf{U}}_t)$ and $\mathbf{u}_0 := \mathrm{vec}(\underline{\mathbf{U}}_0)$ and considering the Cartesian product Laplacian:

$$\mathbf{L}_\diamond = \sum_{p=1}^{P} \mathbf{I}_{N_1} \otimes \cdots \otimes \mathbf{I}_{N_{p-1}} \otimes \mathbf{L}_p \otimes \mathbf{I}_{N_{p+1}} \otimes \cdots \otimes \mathbf{I}_{N_P}. \tag{23}$$

So, we have:

$$\mathbf{u}_t = \cos(t\mathbf{L}_\diamond)\mathbf{u}_0, \tag{24}$$

and the following steps are needed to conclude the proof:

Step 1: Second derivative in vectorized form

Using the standard derivative formula for the matrix cosine:

$$\frac{d^2}{dt^2}\cos(t\mathbf{L}_\diamond) = -\mathbf{L}_\diamond^2 \cos(t\mathbf{L}_\diamond), \tag{25}$$

we have:

$$\mathbf{u}_t'' = -\mathbf{L}_\diamond^2 \mathbf{u}_t. \tag{26}$$

Step 2: Expand $\mathbf{L}_\diamond^2$

Define the Kronecker sum terms:

$$\mathbf{L}_i^{(i)} := \mathbf{I}_{N_1} \otimes \cdots \otimes \mathbf{I}_{N_{i-1}} \otimes \mathbf{L}_i \otimes \mathbf{I}_{N_{i+1}} \otimes \cdots \otimes \mathbf{I}_{N_P}. \tag{27}$$

Then:

$$\mathbf{L}_\diamond = \sum_{i=1}^{P} \mathbf{L}_i^{(i)}, \qquad \mathbf{L}_\diamond^2 = \sum_{i=1}^{P} (\mathbf{L}_i^{(i)})^2 + 2 \sum_{1 \le i < j \le P} \mathbf{L}_i^{(i)} \mathbf{L}_j^{(j)}. \tag{28}$$

Step 3: Map to tensor mode multiplications

For any tensor $\underline{\mathbf{X}}$:

$$(\mathbf{L}_i^{(i)})^2 \operatorname{vec}(\underline{\mathbf{X}}) = \operatorname{vec}(\underline{\mathbf{X}} \times_i \mathbf{L}_i^2), \qquad \mathbf{L}_i^{(i)} \mathbf{L}_j^{(j)} \operatorname{vec}(\underline{\mathbf{X}}) = \operatorname{vec}(\underline{\mathbf{X}} \times_i \mathbf{L}_i \times_j \mathbf{L}_j). \tag{29}$$

Applying this to $\mathbf{u}_t'' = -\mathbf{L}_\diamond^2 \mathbf{u}_t$ and reshaping into tensor form gives:

$$\frac{\partial^2 \underline{\mathbf{U}}_t}{\partial\, t^2} = -\sum_{i=1}^{P} \underline{\mathbf{U}}_t \times_i \mathbf{L}_i^2 - 2 \sum_{1 \le i < j \le P} \underline{\mathbf{U}}_t \times_i \mathbf{L}_i \times_j \mathbf{L}_j, \tag{30}$$

and the proof is completed.

B Proof of Proposition 3.3

Now, by performing mode-$(P+1)$ unfolding on the tensor $f(\underline{\mathbf{U}})$ in Eq. (31) and relying on Lemma A.1, we have:

$$f(\underline{\mathbf{U}}_l)_{(P+1)} = \sum_{\substack{\boldsymbol{\epsilon} \in \{0,1\}^P \\ |\boldsymbol{\epsilon}| \text{ even}}} (-1)^{|\boldsymbol{\epsilon}|/2}\, \mathbf{W}_l^\top \underline{\mathbf{U}}_{l(P+1)} \left[\downarrow \otimes_{p=1}^{P} g_t^{(\epsilon_p)}(\mathbf{L}_p) \right]^\top. \tag{31}$$

Now, by applying transposition and further simplification, we get:

$$\begin{aligned} [f(\underline{\mathbf{U}}_l)_{(P+1)}]^\top &= \sum_{\substack{\boldsymbol{\epsilon} \in \{0,1\}^P \\ |\boldsymbol{\epsilon}| \text{ even}}} (-1)^{|\boldsymbol{\epsilon}|/2} \downarrow \otimes_{p=1}^{P} g_{t_l}^{(\epsilon_p)}(\mathbf{L}_p)\, [\underline{\mathbf{U}}_{l(P+1)}]^\top \mathbf{W}_l \\ &= \cos(t_l \mathbf{L}_\diamond) [\underline{\mathbf{U}}_{l(P+1)}]^\top \mathbf{W}_l, \end{aligned} \tag{32}$$

which concludes the proof.

C Proof of Proposition 3.6

The proof is presented by induction. Therefore, we first show it is true for the case of $P = 2$ as:

$$\begin{aligned} \tilde{\mathbf{L}}_\diamond &= \tilde{\mathbf{L}}_1 \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \tilde{\mathbf{L}}_2 \\ &= (\mathbf{L}_1 + \mathbf{E}_1) \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes (\mathbf{L}_2 + \mathbf{E}_2) \\ &= \overbrace{(\mathbf{L}_1 \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \mathbf{L}_2)}^{\mathbf{L}_\diamond} + \overbrace{(\mathbf{E}_1 \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \mathbf{E}_2)}^{\mathbf{E}} \\ &= \mathbf{L} + \mathbf{E}. \end{aligned} \tag{33}$$

Then, by assuming the theorem holds for the case of $P = K$ and the definitions of $\tilde{\mathbf{L}}_K' := \oplus_{p=1}^{K} \tilde{\mathbf{L}}_p$ and $\tilde{\mathbf{L}}_K' := \oplus_{p=1}^{K} \tilde{\mathbf{L}}_p$, we next show it also holds for $P = K + 1$ as follows:

$$\begin{aligned} \tilde{\mathbf{L}} = \oplus_{p=1}^{K+1} \tilde{\mathbf{L}}_p &= \overbrace{(\oplus_{p=1}^{K} \tilde{\mathbf{L}}_p)}^{\tilde{\mathbf{L}}_K'} \oplus \tilde{\mathbf{L}}_{K+1} = \tilde{\mathbf{L}}_K' \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \tilde{\mathbf{L}}_{K+1} \\ &= (\mathbf{L}_K' + \mathbf{E}_K') \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes (\mathbf{L}_{K+1} + \mathbf{E}_{K+1}) \\ &= \overbrace{(\mathbf{L}_K' \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \mathbf{L}_{K+1})}^{\mathbf{L} = \oplus_{p=1}^{K+1} \mathbf{L}_p} + \overbrace{(\mathbf{E}_K' \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \mathbf{E}_{K+1})}^{\mathbf{E} = \oplus_{p=1}^{K+1} \mathbf{E}_p} \\ &= \mathbf{L} + \mathbf{E}. \end{aligned} \tag{34}$$

To prove the upper bound of the norm of $\mathbf{E}$, in a similar approach to the previous proof, we first prove for the case of $P = 2$ as:

$$\begin{aligned} |||\mathbf{E}||| = |||\mathbf{E}_1 \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \mathbf{E}_2||| &\leq |||\mathbf{E}_1 \otimes \mathbf{I}_2||| + |||\mathbf{I}_1 \otimes \mathbf{E}_2||| \\ &\leq \overbrace{|||\mathbf{E}_1|||}^{\varepsilon_1} . \overbrace{|||\mathbf{I}_2|||}^{\text{constant}} + \overbrace{|||\mathbf{I}_1|||}^{\text{constant}} . \overbrace{|||\mathbf{E}_2|||}^{\varepsilon_2} = \mathcal{O}(\varepsilon_1 + \varepsilon_2). \end{aligned} \tag{35}$$

Using a similar approach to the proof in (34), we assume the theorem holds for $P = K$ as $|||\mathbf{E}||| \leq \mathcal{O}(\sum_{p=1}^{K} \varepsilon_p)$, and, based on this information, then, we prove the theorem for the case of $P = K + 1$ as follows:

$$\begin{aligned} |||\mathbf{E}||| = |||\mathbf{E}'_K \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \mathbf{E}_{K+1}||| &\leq |||\mathbf{E}'_K \otimes \mathbf{I}_2||| + |||\mathbf{I}_1 \otimes \mathbf{E}_{K+1}||| \\ &\leq \overbrace{|||\mathbf{E}'_K|||}^{\sum_{p=1}^{K} \varepsilon_p} . \overbrace{|||\mathbf{I}_2|||}^{\text{constant}} + \overbrace{|||\mathbf{I}_1|||}^{\text{constant}} . \overbrace{|||\mathbf{E}_{K+1}|||}^{\varepsilon_{K+1}} = \mathcal{O}\left(\sum_{p=1}^{K+1} \varepsilon_p\right), \end{aligned} \tag{36}$$

which concludes the proof.

D Proof of Theorem 3.7

Due to the filtering approach $\varphi(u, t) = \cos(t\mathbf{L}_\diamond)\varphi(u, 0)$, we only need to bound $\|\cos(t\tilde{\mathbf{L}}_\diamond) - \cos(t\mathbf{L}_\diamond)\|$, where $\tilde{\mathbf{L}} - \mathbf{L} = \mathbf{E}$. The following Lemma describes the properties of Lipschitz continuity for cosine operators.

Lemma D.1. *Let $\mathbf{A}, \mathbf{B}$ be Hermitian matrices and let $t \geq 0$. Then:*

$$\|\cos(t\mathbf{A}) - \cos(t\mathbf{B})\| \leq t\,\|\mathbf{A} - \mathbf{B}\|.$$

Moreover, the constant t is optimal: one cannot replace t by a smaller constant valid for all Hermitian $\mathbf{A}, \mathbf{B}$.

Proof. Using $\cos(\mathbf{X}) = \frac{1}{2}(e^{\mathrm{i}\mathbf{X}} + e^{-\mathrm{i}\mathbf{X}})$ and the Duhamel identity:

$$e^{\mathrm{i}t\mathbf{A}} - e^{\mathrm{i}t\mathbf{B}} = \mathrm{i} \int_0^t e^{\mathrm{i}(t-s)\mathbf{A}}(\mathbf{A} - \mathbf{B})e^{\mathrm{i}s\mathbf{B}}\, ds,$$

we obtain, since $e^{\mathrm{i}\mathbf{A}}$ and $e^{\mathrm{i}\mathbf{B}}$ are unitary,

$$\|e^{\mathrm{i}t\mathbf{A}} - e^{\mathrm{i}t\mathbf{B}}\| \leq \int_0^t \|\mathbf{A} - \mathbf{B}\|\, ds = t\|\mathbf{A} - \mathbf{B}\|.$$

The same bound holds for $e^{-\mathrm{i}t\mathbf{A}} - e^{-\mathrm{i}t\mathbf{B}}$. Averaging yields:

$$\|\cos(t\mathbf{A}) - \cos(t\mathbf{B})\| \leq \tfrac{1}{2}\|e^{\mathrm{i}t\mathbf{A}} - e^{\mathrm{i}t\mathbf{B}}\| + \tfrac{1}{2}\|e^{-\mathrm{i}t\mathbf{A}} - e^{-\mathrm{i}t\mathbf{B}}\| \leq t\|\mathbf{A} - \mathbf{B}\|.$$

For optimality, note that restricting to commuting (diagonal) matrices reduces the inequality to the scalar Lipschitz bound $|\cos(t\lambda) - \cos(t\mu)| \leq C|\lambda - \mu|$, which forces $C \geq \sup_x |\frac{d\cos(tx)}{dx}| = \sup_x |(-t\sin(tx))| = t$. □

Therefore, using Lemma D.1 and Proposition 3.6, one can write:

$$|||\varphi(u, t) - \tilde{\varphi}(u, t)||| \leq t\,.|||\varphi(u, 0)|||.\,|||\mathbf{E}||| \leq t\,|||\varphi(u, 0)|||\,\left(\sum_{p=1}^{P} \varepsilon_p\right) = \sum_{p=1}^{P} \mathcal{O}(\varepsilon_p). \tag{37}$$

E Proof of Theorem 3.9

First, Note that Using $\tilde{\mathbf{x}}$ as the Graph Fourier Transform (GFT) [16] of $\mathbf{x}$ w.r.t. $\hat{\mathbf{L}}$ (with eigenvalues $\{\lambda_i\}_{i=1}^{N}$), one can write [20]:

$$E(\mathbf{x}) = \mathbf{x}^\top \hat{\mathbf{L}}\mathbf{x} = \sum_{i=1}^{N} \lambda_i \tilde{x}_i^2. \tag{38}$$

Next, by defining λ as the smallest non-zero eigenvalue of the Laplacian $\hat{\mathbf{L}}$, the following lemma characterizes the over-smoothing aspects of applying a heat kernel in the simplest case.

Lemma E.1. *We have:*

$$E(\cos(t\hat{\mathbf{L}})\mathbf{x}) \leq \cos^2(t\lambda_\phi)E(\mathbf{x}), \tag{39}$$

where $\cos^2(t\lambda_\phi) = \max_i \cos^2(t\lambda_i)$.

Proof. By considering the EVD forms of $\hat{\mathbf{L}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ and $\cos(t\hat{\mathbf{L}}) = \mathbf{V}\cos(t\mathbf{\Lambda})\mathbf{V}^\top$, one can write:

$$\begin{aligned} &E(\cos(t\hat{\mathbf{L}})\mathbf{x}) \\ &= \mathbf{x}^\top \overbrace{\cos(t\hat{\mathbf{L}})^\top}^{\mathbf{V}\cos(t\mathbf{\Lambda})\mathbf{V}^\top} \underbrace{\hat{\mathbf{L}}}_{\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top} \overbrace{\cos(t\hat{\mathbf{L}})}^{\mathbf{V}\cos(t\hat{\mathbf{\Lambda}})\mathbf{V}^\top} \mathbf{x} = \sum_{i=1}^{N} \lambda_i \tilde{x}_i^2 \cos^2(t\lambda_i) \\ &\leq \cos^2(t\lambda_\phi)\left(\sum_{i=1}^{N} \lambda_i \tilde{x}_i^2\right) = \cos^2(t\lambda_\phi)E(\mathbf{x}). \end{aligned} \tag{40}$$

Note that in the above proof, we ruled out the zero eigenvalues since they are useless in analyzing the Dirichlet energy. □

Then by considering the following lemmas from [20]:

Lemma E.2. *(Lemma 3.2 in [20]).* $E(\mathbf{X}\mathbf{W}) \leq \|\mathbf{W}^\top\|_2^2 E(\mathbf{X})$.

Lemma E.3. *(Lemma 3.3 in [20]). For ReLU and Leaky-ReLU nonlinearities* $E(\sigma(\mathbf{X})) \leq E(\mathbf{X})$.

Lemma E.4 (Closed form for a cosine of a sum)**.** *For any integer* $P \geq 1$ *and any real numbers* $\lambda_1, \ldots, \lambda_P \in \mathbb{R}$,

$$\cos\left(\sum_{p=1}^{P} \lambda_p\right) = \sum_{\substack{S \subseteq \{1,\ldots,P\} \\ |S| \text{ even}}} (-1)^{|S|/2} \prod_{p \in S} \sin(\lambda_p) \prod_{p \notin S} \cos(\lambda_p) \tag{41}$$

Proof. We set our proof by induction using the two-angle formula. We prove (41) for all $P \in \mathbb{N}$. For $P = 1$, the only even subset is $\varnothing$, giving $\cos(\lambda_1)$. Assume (41) holds for some P and let $L_P = \sum_{p=1}^{P} \lambda_p$. Using $\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y)$,

$$\cos(L_P + \lambda_{P+1}) = \cos(L_P)\cos(\lambda_{P+1}) - \sin(L_P)\sin(\lambda_{P+1}). \tag{42}$$

The companion identity is

$$\sin(L_P) = \sum_{\substack{S \subseteq \{1,\ldots,P\} \\ |S| \text{ odd}}} (-1)^{(|S|-1)/2} \Big(\prod_{p \in S} \sin(\lambda_p)\Big)\Big(\prod_{p \notin S} \cos(\lambda_p)\Big).$$

Substituting the two expansions into (42) and distributing shows that all even-cardinality subsets $T \subseteq \{1, \ldots, P+1\}$ occur with coefficient $(-1)^{|T|/2}$, yielding (41) with $P+1$ in place of P. Thus the result holds for all P by induction. □

Therefore, by combining the previous concepts, for l layers, it can be said that:

$$\begin{aligned} &E(\mathbf{X}^{(l)}) \leq (s\cos^2(t\lambda_\phi^\diamond))^l E(\mathbf{X}), \\ &\text{where } \cos(t\,\lambda_\phi^\diamond) = \sum_{\substack{S \subseteq \{1,\ldots,P\} \\ |S| \text{ even}}} (-1)^{|S|/2} \prod_{p \in S} \sin(t\,\lambda_{i_p}^{(p)}) \prod_{p \notin S} \cos(t\,\lambda_{i_p}^{(p)}). \end{aligned} \tag{43}$$

Here, $\lambda_{i_p}^{(p)}$ is the i_p-th eigenvalue of the p-th normalized factor graph such that $\cos^2(t\lambda_\phi^\diamond) = \max_i \cos^2(t\lambda_i^\diamond)$, where $\lambda_i^\diamond$ is the i-th non-zero eigenvalue of $\hat{\mathbf{L}}_\diamond$ and $\lambda_\phi^\diamond = \sum_{p=1}^{P} \lambda_{i_p}^{(p)}$.

So, $E(\mathbf{X}^{(l)})$ exponentially converges to 0, when $\lim_{l\to\infty}(s\cos^2(t\lambda_\phi))^l = 0$, i.e., $s\cos^2(t\lambda_\phi) < 1$:

$$t \in \bigcup_{k \in \mathbb{Z}} \frac{1}{2\lambda_\phi^\diamond}\left(2k\pi + \arccos\left(\frac{2}{s} - 1\right),\ 2(k+1)\pi - \arccos\left(\frac{2}{s} - 1\right)\right). \tag{44}$$