ADASPEC: ADAPTIVE SPECTRUM FOR ENHANCED NODE DISTINGUISHABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Spectral Graph Neural Networks (GNNs) achieve strong performance in node classification, yet their node distinguishability remains poorly understood. We analyze how graph matrices and node features jointly influence node distinguishability. Further, we derive a theoretical lower bound on the number of distinguishable nodes, which is governed by two key factors: distinct eigenvalues in the graph matrix and nonzero frequency components of node features in the eigenbasis. Based on these insights, we propose AdaSpec, an adaptive graph matrix generation module that enhances node distinguishability of spectral GNNs without increasing the order of computational complexity. We prove that AdaSpec preserves permutation equivariance, ensuring that reordering the graph nodes results in a corresponding reordering of the node embeddings. Experiments across eighteen benchmark datasets validate AdaSpec's effectiveness in improving node distinguishability of spectral GNNs.

1 INTRODUCTION

Graph Neural Networks (GNNs) have become increasingly popular for graph learning tasks due to their strong performance in tasks such as graph and node classification (Kipf & Welling, 2017; Xu et al., 2019; He et al., 2021; Wang & Zhang, 2022; Qin et al., 2025). Among the various GNN models, spectral GNNs represent a prominent class that transforms graph signals into the spectral domain, enabling graph filters to process information for downstream tasks. Although numerous spectral GNN variants have been proposed, their node distinguishability remains insufficiently understood. These models typically utilize different graph matrices, such as the normalized adjacency matrix or the normalized Laplacian. Further, within a given spectral GNN, the distribution of node features across the graph plays a crucial role in model performance (He et al., 2022b; Platonov et al., 2023). To the best of our knowledge, no existing work has systematically analyzed the interaction between the graph matrix and node features in determining node distinguishability in spectral GNNs.

Spectral GNNs with state-of-the-art performance generally follow the form:

$$\Psi(M, X) = g_\Theta(M) f_W(X), \tag{1}$$

where $M \in \mathbb{R}^{n\times n}$ represents the graph matrix (such as the Laplacian or adjacency matrix), $X \in \mathbb{R}^{n\times h}$ denotes the node feature matrix, $g_\Theta(M) = \sum_{k=0}^{K} \theta_k T_k(M)$ is the graph convolution function parameterized by $\Theta = \{\theta_k\}_{k=0}^{K}$, and $T_k(\cdot)$ denotes the k-th polynomial basis. The term $f_W(X)$ represents the feature transformation function parameterized by W. Spectral GNNs learn meaningful node features by optimizing W, projecting them into the spectral domain. By adjusting Θ, spectral GNNs filter out unnecessary information and enhance useful information for downstream tasks.

While this formulation illustrates how spectral GNNs process node features through graph convolution, their capacity for node distinguishability remains inadequately understood. This leads to a fundamental question: how does the interaction between the graph matrix M and the node features X projected into the spectral domain affect the node distinguishability of spectral GNNs? In this work, we demonstrate that node distinguishability is influenced by the eigenvalue multiplicity and the missing frequency components of node features in the eigenbasis of the graph matrix. Further, we derive a theoretical lower bound on the number of nodes that can be distinguished by spectral GNNs, given a specific graph matrix and node features.

Motivated by our theoretical analysis of node distinguishability, we introduce AdaSpec, an adaptive graph matrix generation module that optimizes the graph matrix to maximize its lower bound on node distinguishability. Designed as a plug-in, AdaSpec can be seamlessly integrated

into any spectral GNN to enhance node distinguishability. Moreover, spectral GNNs augmented with AdaSpec preserve permutation equivariance, ensuring that reordering graph nodes results in a corresponding reordering of node embeddings. Finally, AdaSpec maintains the graph's connectivity, guaranteeing that the learned embeddings accurately reflect the underlying graph structure.

We evaluate our approach on eighteen benchmark node classification datasets, covering a range of small- and large-scale graphs with both homophilic and heterophilic structures in Section 6. Spectral GNNs with AdaSpec achieve notable performance improvements on heterophilic graphs, while maintaining or slightly improving accuracy on homophilic ones. These results validate the effectiveness of AdaSpec in boosting node distinguishability. Additionally, experimental results show that the order of time complexity of spectral GNNs with and without AdaSpec are the same.

2 Related Works

Spectral GNNs. Spectral GNNs perform graph convolution by applying filters in the spectral domain for representation learning. Based on the design of their graph filters, spectral GNNs can be categorized into polynomial (He et al., 2022a; 2021) and rational types (Levie et al., 2019; Bianchi et al., 2021; Li et al., 2025). Polynomial graph filters are computationally efficient and localized in the vertex domain (Hammond et al., 2009; Defferrard et al., 2016), and this paper focuses on their analysis. Recent studies primarily investigate how different polynomial bases affect spectral GNN performance, for instance, ChebNet, ChebNetII, JacobiConv, BernNet, GPRGNN and GLN (Defferrard et al., 2016; He et al., 2022a; Wang & Zhang, 2022; He et al., 2021; Chien et al., 2021; Li & Wang, 2024). Further, FavardGNN, UniFilter and PolyCF learn polynomial bases that adapt to different graph structures (Guo & Wei, 2023; Huang et al., 2024; Qin et al., 2025).

Above spectral GNNs use fixed graph matrices like normalized adjacency or Laplacian matrices. While research has focused on effect of polynomial bases on performance of spectral GNNs, we demonstrate the critical role of the graph matrix. We analyze how the interaction between the graph matrix and node features affects spectral GNN performance. Further, we propose AdaSpec, a graph matrix generation module to enhance the performance of spectral GNNs.

Expressive Power of Spectral GNNs. The expressive power of GNNs in graph classification has been extensively analyzed through the Weisfeiler-Lehman (WL) test (Li & Leskovec, 2022; Zhang et al., 2023; Jin et al., 2025), a family of algorithms designed to determine graph isomorphism (Weisfeiler & Leman, 1968). In contrast, the expressive power of GNNs for node classification remains less explored. The expressive capacity of linear spectral GNNs has been analyzed via the uniform approximation theorem in (Wang & Zhang, 2022), which shows that when the graph matrix has no repeated eigenvalues and node features span all frequency components, the model can approximate any one-dimensional function. However, these conditions rarely hold in real-world graphs, where symmetric structures are common and node features are often sparse. As shown in Figure 2, eigenvalue multiplicity and missing frequency components frequently occur in practice. An eigenvalue correction method was proposed in (Lu et al., 2024) to enhance the expressiveness of spectral GNNs, building on the analysis in (Wang & Zhang, 2022). However, this method does not ensure permutation equivariance, leading to node embeddings that depend on node ordering, which is undesirable and theoretically unsound.

Our work investigates the expressive power of spectral GNNs from the perspective of node distinguishability. We extend the understanding of how the interaction between the graph matrix and node features influences node distinguishability in spectral GNNs. Notably, our analysis goes beyond linear GNNs by incorporating nonlinear feature transformations f_W. Moreover, we rigorously establish a theoretical lower bound on the number of distinguishable nodes in spectral GNNs.

Graph Rewiring. Another line of research focuses on improving the performance of GNNs through graph rewiring techniques, which modify the graph topology. Early methods include DropEdge and EDGEWIRE, which randomly or uses degree-preserving strategy to remove edges to alleviate over-smoothing (Rong et al., 2020; Chan & Akoglu, 2016). Curvature-based approaches (Topping et al., 2022) adjust connectivity using discrete Ricci curvature to combat over-squashing, while locality-aware strategies preserve structures efficiency (Barbero et al., 2024). More recent methods include DiffWire, a differentiable and parameter-free approach guided by the Lovász bound (Arnaiz-Rodrıéguez et al., 2022); FoSR, which improves spectral expansion (Karhadkar et al., 2023); and GPER, selecting edges based on effective resistance to enhance information flow (Shen et al., 2024).

While graph rewiring methods offer valuable insights into improving GNN performance, their objectives and underlying mechanisms differ fundamentally from ours. Graph rewiring addresses structural issues by modifying graph topology in the spatial domain as a preprocessing step. In contrast, our method enhances node distinguishability in the spectral domain through an adaptive graph matrix generation module that trains end-to-end with spectral GNNs.

3 PRELIMINARIES

Let $G = (\mathcal{V}, \mathcal{E}, X)$ denote an undirected, simple graph, where $\mathcal{V}$ is the set of nodes with cardinality $|\mathcal{V}| = n$, $\mathcal{E}$ is the set of edges, and $X \in \mathbb{R}^{n\times h}$ is the node feature matrix. For each node $v \in \mathcal{V}$, $X(v) \in \mathbb{R}^h$ denotes its associated feature vector. The graph structure is represented by the adjacency matrix $A \in \{0,1\}^{n\times n}$, where $A_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$, and 0 otherwise. The degree matrix $D \in \mathbb{R}^{n\times n}$ is diagonal with entries D_{ii} equal to the degree of node v_i. The normalized adjacency matrix is defined as $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. The normalized graph Laplacian is given by $\tilde{L} = I - \tilde{A}$, where $I \in \mathbb{R}^{n\times n}$ is the identity matrix.

Two nodes u and v in an undirected graph G are *structurally equivalent* $s_u \sim s_v$ if they share exactly the same neighbors; formally, for every other node $w \in \mathcal{V} \setminus \{u, v\}$, $(u, w) \in \mathcal{E} \iff (v, w) \in \mathcal{E}$. In effect, swapping u and v leaves the graph's adjacency relation unchanged.

A *permutation* of the node set $\mathcal{V}$ is a bijection $\pi : \mathcal{V} \to \mathcal{V}$. The set of all permutations on $\mathcal{V}$ forms the symmetric group $\mathrm{Sym}(\mathcal{V})$. An *automorphism* of the graph G is a permutation $\pi \in \mathrm{Sym}(\mathcal{V})$ satisfying the following conditions: (1) edge preservation: $(v, u) \in \mathcal{E} \iff (\pi(v), \pi(u)) \in \mathcal{E}, \quad \forall v, u \in \mathcal{V}$, and (2) feature preservation: $X(\pi(v)) = X(v), \quad \forall v \in \mathcal{V}$. The *automorphism group* of G, denoted $\mathrm{Aut}(G)$, is the set of all such automorphisms.

Two nodes u and v are said to be *isomorphic*, denoted $u \sim v$, if they belong to the same orbit under $\mathrm{Aut}(G)$; that is, there exists a permutation $\pi \in \mathrm{Aut}(G)$ such that $\pi(v) = u$. Otherwise, u and v are *non-isomorphic*.

An important property of functions defined on graphs is *permutation equivariance*, which ensures that the output remains consistent under any reordering of the nodes. Formally,

Definition 3.1 (Permutation Equivariance)**.** Let $\mathcal{G}$ denote the set of graphs. A function $f : \mathcal{G} \to \mathbb{R}^{n\times d}$ is said to be *permutation equivariant* if, for any graph $G \in \mathcal{G}$ and any permutation $\pi \in \mathrm{Sym}(\mathcal{V})$, it holds that

$$f(\pi(G)) = \pi(f(G)),$$

where $\pi(G)$ denotes the graph obtained by permuting the nodes of G according to π, and $\pi(f(G))$ denotes the corresponding permutation of the output of f.

4 NODE DISTINGUISHABILITY OF SPECTRAL GNNS

The node distinguishability of a spectral GNN refers to its ability to distinguish non-isomorphic nodes within graphs. Formally,

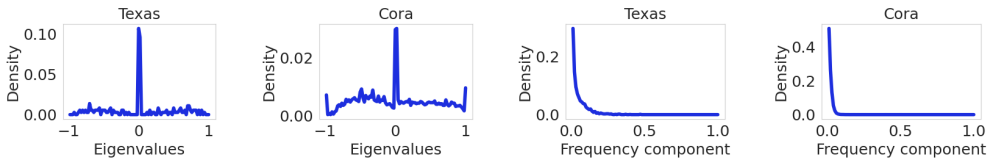
Definition 4.1 (Node Distinguishability)**.** For a spectral GNN with function class $\mathcal{F}$, where each $f \in \mathcal{F} : \mathcal{G} \to \mathbb{R}^{n\times d}$ maps a graph to node representations, node distinguishability refers to the ability to learn a function that assigns distinct representations to non-isomorphic nodes:

$$f(G)_v \neq f(G)_u \quad \text{for all } v, u \in \mathcal{V} \text{ where } v \not\sim u$$

where $f(G)_v$ and $f(G)_u$ denote representations of node v and u. $v \not\sim u$ indicates node u, v are non-isomorphic.

The spectral GNN's node distinguishability capacity that mapping non-isomorphic nodes to distinct representations is fundamentally determined by its function class $\mathcal{F}$. To understand how spectral GNNs of the form given in Equation (1) distinguish nodes, whose input consists of a graph matrix M and a feature matrix X, we begin by formally defining the spectrum of M and the frequency components of X.

Definition 4.2 (Spectrum and Frequency Components)**.** Let $M = U\Lambda U^\top$ be the eigendecomposition of a graph matrix $M \in \mathbb{R}^{n\times n}$, where Λ is a diagonal matrix of eigenvalues and $U = [u_1, \ldots, u_n]$ contains the corresponding eigenvectors. The *spectrum* of M, denoted $\mathrm{spec}(M)$, is the multiset of eigenvalues: $\mathrm{spec}(M) = \{\{\lambda_1, \lambda_2, \ldots, \lambda_n\}\}$, where $\lambda_i = \Lambda_{ii}$. Let support $\mathrm{supp}(\mathrm{spec}(M))$ be



(a) (b) (c) (d)

Figure 2: Eigenvalues and frequency component distributions.

the underlying set of spec(M). Define $d_M = |\text{supp}(\text{spec}(M))|$, which is the number of distinct eigenvalues. Given node features $X \in \mathbb{R}^{n\times h}$, the frequency components in the eigenbasis of M are $\tilde{X} = U^\top X$, where $\tilde{X}_i = u_i^\top X$ is the i-th frequency component. The number of non-zero frequency components is $\|\tilde{X}^{(M)}\|_0 = |\{\tilde{X}_i \mid \tilde{X}_i \neq 0_h\}|$.

The limitations of node distinguishability in spectral GNNs stem from two key factors: Eigenvalue multiplicity of the graph matrix M and the missing of frequency components of node features X when projected onto the eigenbasis of M. In Figure 1, we show that spectral GNNs with a first-order polynomial filter and normalized adjacency matrix $\tilde{A}$ as graph matrix cannot distinguish node 1 and 3. (1) Non-distinguishable nodes can exist when there are missing frequency components that $d_{\tilde{A}} = 5 = n$ but $\|X^{(\tilde{A})}\|_0 = 3 < n$ in Figure 1(a). (2) Non-distinguishable nodes can exist when there are repeated eigenvalues $d_{\tilde{A}} = 3 < n$ even if $\|X^{(\tilde{A})}\|_0 = 5 = n$ in Figure 1(b). Nodes 1 and 3 in both subfigures are non-isomorphic but spectral GNNs yield identical embeddings for them. Hence they are indistinguishable. We provide a theoretical bound on the number of nodes that can be distinguished by spectral GNNs, stated as follows.

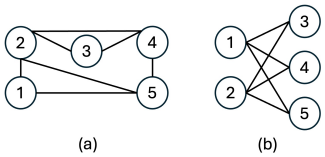


Figure 1: Nodes 1 and 3 cannot be distinguished by spectral GNNs of $K = 1$ with $\tilde{A}$. (a) Missing frequency components: $X = [1, 0, 1, -1, -1]$, $d_{\tilde{A}} = 5$, $\|X^{(\tilde{A})}\|_0 = 3$. (b) Eigenvalue multiplicity: $X = [1, 0, 1, 1, -1]$, $d_{\tilde{A}} = 3$, $\|X^{(\tilde{A})}\|_0 = 5$.

Theorem 4.3. *For $X \neq 0_{n\times n}$, there exist a spectral GNN $\Psi(M, X)$ that can distinguish at least $\min(d_M, \|\tilde{X}^{(M)}\|_0)$ nodes on graph.*

This result provides a fundamental guarantee on the node distinguishability of spectral GNNs. The lower bound depends on both the number of distinct eigenvalues d_M and the number of non-zero frequency components $\|\tilde{X}^{(M)}\|_0$, which together characterize the alignment between the graph matrix M and the node features X. When multiple eigenvectors share the same eigenvalue, the graph filter g_Θ applies identical transformations to them, preventing from distinguishing different structural patterns. Similarly, if node features lack frequency components corresponding to certain eigenvectors, structural differences captured by those eigenvectors become invisible in embeddings. This has practical implications: increasing distinct eigenvlaue number d_M and non-zero frequency components of X in the eigenbasis of M improves the theoretical guarantee on the lower bound of number of distinguishable nodes, offering a clear direction for enhancing the expressive power of spectral GNNs.

In real-world graphs, we observe that eigenvalue multiplicity and missing frequency component are very common.

Observation I (Eigenvalues of Multiplicity.) The normalized graph adjacency matrix $\tilde{A} = D^{-1/2}AD^{-1/2}$ often contains eigenvalues with multiplicities greater than one and the eigenvalue zero has largest multiplicity.

We illustrate the eigenvalue distribution of the normalized graph adjacency matrix for the Texas and Cora datasets in Figure 2(a-b). Additional eigenvalue distributions for various other real-world datasets are provided in Figure 3 (Appendix). This phenomenon is also observed in (Lim et al., 2023). Graph symmetry, repeated substructures often lead to repeated eigenvalues in the normalized adjacency matrix and reduce its rank. Real-world graphs also tend to be sparse due to many low-degree nodes, further lowering the rank. Since the rank of a real symmetric matrix equals the number of non-zero eigenvalues, low-rank matrices imply high multiplicity of the zero eigenvalue.

Node features in connected real-world graphs are sampled independently of the graph structure. For instance, in citation networks (such as Cora and PubMed), node features are the textual content of

papers, which are collected independently of the graph structure. Thus, graph signals are not aligned with the graph's eigenvectors. We have below observations.

Observation II (Missing Frequency Components.) Many frequency components of graph signal (node feature) is zero in the eigenbasis of normalized graph adjacency matrix $\tilde{A}$.

We illustrate the distribution of frequency components for Texas and Cora in Figure 2(c-d), where most components are zero. Additional results for other real-world datasets are provided in Figure 4 (Appendix). Zero frequency component means that the frequency component in the direction of corresponding eigenvectors is missing. Real-world node features are often either smooth or oscillatory, containing only low or high-frequency components, leading to many others to be zero or negligible. Additionally, features are typically sparse, with only k non-zero entries that $k \ll n$. When projected onto the eigenbasis, each component scales as $O(k/\sqrt{n})$. As $n \to \infty$, the proportion of non-zero frequency components tends toward zero.

Based on above observations and Theorem 4.3, we propose AdaSpec to enhance the node distinguishability of spectral GNNs.

5 AdaSpec

AdaSpec generates a graph matrix that adapts to both the graph structure and node features, enabling it to serve as a plug-in module for any spectral GNN $\Psi(M, X)$ of the form in Equation (1). The spectral GNN augmented with AdaSpec is defined as:

$$\Psi^{+}(A, X) = g_{\Theta}(\Omega(A, X))f_{W}(X), \tag{2}$$

where Ω maps the adjacency matrix A and node features X to a new graph matrix. The functions g_{Θ} and $f_{W}(X)$ remain the same as those in $\Psi(M, X)$.

AdaSpec enables $\Psi^{+}(A, X)$ to capture richer interactions between graph structure and node features, which are not possible using fixed matrices in classic spectral GNNs $\Psi(M, X)$. To ensure permutation equivariance of node embeddings, the generated graph matrix $M = \Omega(A, X)$ must satisfy two key properties: (1) M commutes with $\mathrm{Aut}(G)$: $P_{\sigma}M = MP_{\sigma}, \forall \sigma \in \mathrm{Aut}(G)$ where P_{σ} is the permutation matrix corresponding to the automorphism σ; (2) M preserves edge connectivity: $M_{ij} \neq 0 \Leftrightarrow e_{ij} \in \mathcal{E}$ and $M_{ij} = 0 \Leftrightarrow e_{ij} \notin \mathcal{E}$. Thus, we design $\Omega(A, X)$ as

$$\Omega(A, X) = \Omega_{D}(A) + \alpha_{1}\Omega_{S}(A) + \alpha_{2}\Omega_{F}(X) \tag{3}$$

where $\Omega_{D}(A)$ is designed to increase the number of distinct eigenvalues, $\Omega_{S}(A)$ aims to reduce the multiplicity of zero eigenvalues, and $\Omega_{F}(X)$ is designed to decrease missing frequency components of X. The hyperparameters α_{1}, α_{2} control the eigenvalue range for stable training.

5.1 Increase Distinct Eigenvalues

According to Theorem 4.3, increasing the number of distinct eigenvalues of the graph matrix can raise the lower bound of number of nodes distinguished by a spectral GNN, thereby increasing its node distinguishability. To achieve this, the term $\Omega_{D}(A)$ in AdaSpec is designed as follows:

$$\Omega_{D}(A) = (D + B)^{-1/2}(A + B)(D + B)^{-1/2},$$

where A and D are the graph adjacency matrix and the degree matrix, respectively, and $B = \mathrm{diag}(b)$ is a learnable diagonal matrix with non-negative elements.

The diagonal element of B is initialized as $b_{u} = 1/D_{uu}$, ensuring nodes with the same degree start with the same bias. For isomorphic nodes $u \sim v$, we have $b_{u} = b_{v}$ throughout training; for $u \not\sim v$, training yields $b_{u} \neq b_{v}$. This initialization preserves permutation equivariance of $\Psi^{+}(A, X)$, as shown in Proposition 5.5. Adding B to A introduces node-specific flexibility, enabling $A + B$ and $D + B$ to adapt to graphs. This enhances node distinguishability by allowing structurally equivalent but feature different nodes to play distinct roles. For two non-isomorphic nodes u, v that $u \not\sim v$, if $s_{u} \sim s_{v}$ but $X(u) \neq X(v)$, introducing different biases $b_{u} \neq b_{v}$ breaks structure symmetry and reduces eigenvalue multiplicity. Intuitively, B modifies the self-loop strength, altering information flow from the node itself. We later provide theoretical justification that this increases the number of distinct eigenvalues.

Theorem 5.1 (Increased Distinct Eigenvalues)**.** *Given a graph G with the adjacency matrix A, and the degree matrix D, we have:*

$$d_{\Omega_{D}(A)} \geq d_{\tilde{A}}$$

This indicates that the lower bound of the number of distinguishable nodes for spectral GNNs using Ω_D is greater than or equal to that for those using $\tilde{A}$, according to Theorem 4.3.

5.2 Shifts Eigenvalues From Zero

The presence of zero eigenvalues can hinder node distinguishability. To mitigate this, we design a graph matrix transformation that shifts eigenvalues away from zero:

$$\Omega_S(A) = I.$$

Several approaches exist for designing Ω_S; we choose the identity matrix because adding it to any matrix shifts the eigenvalues while preserving the eigenvectors. This ensures minimal alteration to the original matrix.

Adding term $\epsilon\Omega_S$ to any matrix C can reduce the number of zero eigenvalues. As all eigenvalues of C add the same scalar ϵ, distinct eigenvalues remain distinct after addition. As all eigenvectors of C stays the same, so the number of non-zero frequency component of node feature stays the same.

5.3 Increase Frequency Components

We can increase the number of non-zero frequency component to the node distinguishability of spectral GNNs. Given a node feature matrix X, we design a matrix Ω_F that adapts to X to increase the frequency components:

$$\Omega_F(X) = \sum_{i=1}^{h} \frac{X_{:i}X_{:i}^\top}{\|X_{:i}\|_F^2} \circ A \tag{4}$$

where $\circ$ denotes the Hadamard product.

By dividing by the Frobenius norm $\|X_{:i}\|_F^2$, features with larger magnitudes don't dominate the transformation. The Hadamard product with the adjacency matrix A preserves the graph's original structure. We prove in theory that for any symmetric matrix C, adding $\epsilon\Omega_F(X)$ can increase non-zero frequency components.

Theorem 5.2 (Non-Decreasing Frequency Components)**.** *For a real symmetric matrix $C \in \mathbb{R}^{n\times n}$ with orthonormal basis $\{u_r\}_{r\in[n]}$. Under Condition 5.3, the following holds for index $i \in [h]$:*

$$\|\tilde{X}_{:i}^{(C+\epsilon\Omega_F)}\|_0 > \|\tilde{X}_{:i}^{(C)}\|_0$$

where ϵ is a non-zero constant.

Condition 5.3 (Non-zero feature projections)**.** Let $C \in \mathbb{R}^{n\times n}$ be a real symmetric matrix with orthonormal eigenbasis $\{u_r\}_{r=1}^n$. There exist two column node feature vectors $X_{:i}$ and $X_{:l}$ with $i, l \in [h]$ and $i \neq l$ such that $u_k^\top X_{:i} \neq 0$, $u_k^\top X_{:l} \neq 0$, and $u_j^\top X_{:l} \neq 0$ for some indices $k, j \in [n]$.

Condition 5.3 are naturally satisfied in most real-world graph datasets. This condition requires that node features have non-zero projections onto certain eigenvectors of the graph matrix. Natural heterogeneity in node features makes it likely that different nodes will have diverse nonzero projections onto eigenvectors, even with sparse features. Additionally, while feature correlation exists, real-world graph typically varies a lot along certain dimensions, satisfying our non-zero projection condition. Therefore, incorporating $\Omega_F(X)$ ensures that the number of non-zero frequency components of node features is increased in real-world graphs.

In summary, each component of $\Omega(A, X)$ either increases the number of distinct eigenvalues or the number of non-zero frequency components of the node features in the eigenbasis of the graph matrix. By Theorem 4.3, this leads to a higher lower bound on the number of distinguishable nodes, thereby enhancing node distinguishability. We show properties of our design $\Omega(A, X)$ as below.

Theorem 5.4. *For a graph G, the learnable matrix $\Omega(A, X)$ is commutative with $\mathrm{Aut}(G)$ and preserves edge connectivity.*

As $\Omega(A, X)$ satisfies desirable properties, it ensures that the augmented spectral GNNs $\Psi^+(A, X)$ with AdaSpec remains permutation equivariant.

Proposition 5.5. *When f_W is permutation equivariant, spectral GNNs $\Psi^+(A, X)$ augmented with AdaSpec is permutation equivariant.*

Theorem 5.4 and Proposition 5.5 ensures that for spectral GNNs $\Psi^+(A, X)$, reordering the graph nodes results in a corresponding reordering of node embeddings. AdaSpec can be combined with any spectral GNNs to enhance their node distinguishability.

5.4 Time Complexity Analysis

The time complexity of classic spectral GNNs $\Psi(M, X)$ and $\Psi^+(A, X)$ augmented with AdaSpec is in the same order in both forward and backward propagation. $\Omega_F(X)$ in AdaSpec will increase the pre-computing time, but it needs to be computed only once. We list the time complexity in Table 1.

The time complexity can be analyzed in two main phases: pre-computation and forward/backward propagation. During pre-computation, graph matrix normalization requires $O(|\mathcal{V}| + |\mathcal{E}|)$ operations such as graph adjacency matrix normalization. $\Omega_F(X)$ in $\Psi^+(A, X)$ requires an additional $O(h(|\mathcal{V}|+|\mathcal{E}|))$ where computation is efficiently limited to non-zero entries in the adjacency matrix. Thus, the one-off pre-computing of $\Psi^+(A, X)$ scales linearly in the size of graph and node feature dimension.

For forward and backward propagation, the feature transformation step $f_W(X)$ incurs a complexity of $O(|W|h)$, while graph convolution g_Θ requires $O(KT|\mathcal{E}|)$ operations when $T_k(M)$ is computed recursively, such as in ChebNet, JacobiConv. Although $\Psi^+(A, X)$ requires additional computation of $\Omega(A, X)$ during each forward pass and gradient calculation for matrix B during backpropagation at a cost of $O(|\mathcal{V}| + |\mathcal{E}|)$, this does not change the overall asymptotic complexity.

6 Experiments

We design our experiments to investigate the following research questions: (1) **Q1:** To what extent does AdaSpec generate task-adaptive graph matrices that enhance node distinguishability in spectral GNNs? (2) **Q2:** What is the contribution of each component within AdaSpec to overall performance? (3) **Q3:** How does AdaSpec affect the spectral properties of the graph matrix, particularly in terms of increasing the number of distinct eigenvalues? (4) **Q4:** What is the computational overhead introduced by integrating AdaSpec into spectral GNNs during training?

Experimental Setup. We conduct experiments on eighteen benchmark datasets for node classification to verify the effectiveness of AdaSpec. Datasets includes: six small heterophilic graphs (Texas, Wisconsin, Actor, Chameleon, Squirrel, Cornell), five large heterophilic graphs (Roman_Empire, Amazon_Ratings, Minesweeper, Tolokers, Questions) and seven homophilic graphs (Citeseer, Pubmed, Cora, Computers, Photo, Coauthor-CS, Coauthor-Physics). Statistics of datasets, details about the baselines, and the setting of hyperparameters are included in Appendix B. For each dataset, we follow (Chien et al., 2021; He et al., 2022a) and use sparse splitting that nodes are randomly divided into training/validation/testing with ratios of $2.5\%/2.5\%/95\%$, respectively. Notably, for Citeseer, Pubmed, and Cora datasets, 20 nodes per class are for training, 500 nodes for validation, and 1,000 nodes for testing.

We chose five popular spectral GNNs as our baselines: ChebNet (Defferrard et al., 2016), GPRGNN (Chien et al., 2021), BernNet (He et al., 2021), JacobiConv (Wang & Zhang, 2022), and ChebNetII (He et al., 2022a), and compare their performances augmented with AdaSpec and with fixed graph matrix across all datasets. For each spectral GNN, we use GNN (O) to denote the original model and GNN (M) to denote the spectral GNNs augmented by AdaSpec, with $\Delta \uparrow$ indicating the performance improvement.

Effectiveness of AdaSpec. We present the node classification performance with and without the AdaSpec on all small heterophilic datasets and a subset of large heterophilic datasets in Table 2. The Minesweeper and Question datasets are particularly challenging to classify, as their label informativeness (i.e., the mutual information between the labels of the central node and its neighbors) is zero (Platonov et al., 2023). The complete experimental results are in Table 9 (Appendix). Results on homophilic graphs are shown in Table 3 .

Spectral GNNs	Parameter Count	Pre-computing Complexity	Forward/Backward Complexity
$\Psi(M, X)$	$1 + K$	$O(\|\mathcal{V}\| + \|\mathcal{E}\|)$	$O(KT\|\mathcal{E}\| + \|\mathcal{V}\|\|W\|)$
$\Psi^+(A, X)$	$1 + K + \|\mathcal{V}\|$	$O(h(\|\mathcal{V}\| + \|\mathcal{E}\|))$	$O(KT\|\mathcal{E}\| + \|\mathcal{V}\|\|W\|)$

Table 1: Time complexity comparison of GNNs with and without AdaSpec. $\mathcal{V}$ and $\mathcal{E}$ denotes the node and edge set respectively. h is the node feature dimension. T is the node class number. K is the polynomial order of spectral GNNs.

Model	Texas	Wisconsin	Actor	Chameleon	Squirrel	Cornell	Minesweeper	Questions
ChebNet(O)	38.67±9.31	32.92±7.38	25.15±0.69	29.32±4.13	24.23±3.24	31.33±7.51	86.29±0.2	55.13±0.54
ChebNet(M)	51.16±8.56	33.83±9.38	25.38±0.67	29.73±3.3	23.2±3.94	33.47±7.92	86.7±0.23	55.2±1.52
Δ ↑	+12.49	+0.91	+0.23	+0.41	-1.03	+2.14	+0.41	+0.07
ChebNetII(O)	56.24±1.39	51.5±5.63	29.89±0.68	35.26±3.66	37.19±0.66	39.54±6.88	78.35±0.14	64.13±0.95
ChebNetII(M)	56.76±3.12	52.0±7.75	30.43±1.23	35.62±3.52	36.88±0.69	39.94±7.05	79.1±0.09	65.54±0.7
Δ ↑	+0.52	+0.5	+0.54	+0.36	-0.31	+0.4	+0.75	+1.41
JacobiConv(O)	55.09±5.95	49.0±10.51	32.15±0.77	34.29±3.82	29.29±1.99	38.96±8.79	87.34±0.12	64.72±0.38
JacobiConv(M)	57.4±3.93	52.33±8.88	32.52±0.75	38.16±1.18	31.35±1.68	41.62±10.06	89.13±0.1	65.8±0.18
Δ ↑	+2.31	+3.33	+0.37	+3.87	+2.06	+2.66	+1.79	+1.08
GPRGNN(O)	48.15±4.74	44.25±5.92	30.39±1.24	32.5±2.92	27.7±3.88	34.39±6.88	87.15±0.49	53.14±0.27
GPRGNN(M)	58.27±4.97	53.25±7.21	30.4±1.51	32.82±4.76	27.3±6.03±4.77	36.13±7.52	88.58±0.18	58.19±0.36
Δ ↑	+10.12	+9.0	+0.01	+0.32	-0.4	+1.74	+1.43	+5.05
BernNet(O)	56.19±7.52	49.38±5.75	30.5±1.18	35.35±3.46	33.41±3.42	36.82±10.64	76.54±0.23	64.86±0.37
BernNet(M)	58.9±4.11	51.96±7.84	30.61±0.67	39.61±1.55	34.46±3.52	40.23±5.66	76.95±0.21	65.2±0.31
Δ ↑	+2.71	+2.58	+0.11	+4.26	+1.05	+3.41	+0.41	+0.34

Table 2: Performance of spectral GNNs with/without AdaSpec on heterophilic datasets. ROC AUC is reported on Minesweeper, Questions. Testing accuracy is reported on other datasets. High accuracy and ROC AUC indicate good performance.

Model	Citeseer	Pubmed	Cora	Computers	Photo	Coauthor-CS	Coauthor-Physics
ChebNet(O)	69.21±0.87	75.29±2.34	80.45±1.09	82.64±1.76	91.77±0.32	90.95±0.34	95.03±0.11
ChebNet(M)	68.52±0.86	77.38±1.45	82.26±0.84	85.14±0.89	92.34±0.41	91.54±0.22	94.93±0.09
Δ ↑	-0.69	+2.09	+1.81	+2.5	+0.57	+0.59	-0.1
ChebNetII(O)	69.93±1.15	78.42±1.48	81.64±0.86	84.96±0.97	92.71±0.46	93.08±0.27	95.23±0.1
ChebNetII(M)	69.54±0.9	78.59±1.52	81.97±0.86	84.79±0.83	92.58±0.31	93.11±0.25	95.26±0.11
Δ ↑	-0.39	+0.17	+0.33	-0.17	-0.13	+0.03	+0.03
JacobiConv(O)	70.8±0.7	79.43±1.45	77.15±0.96	85.39±0.95	92.79±0.38	93.33±0.23	95.32±0.15
JacobiConv(M)	70.91±0.66	79.65±1.25	83.52±0.69	84.92±0.92	92.83±0.36	93.27±0.25	95.43±0.11
Δ ↑	+0.11	+0.22	+6.37	-0.47	+0.04	-0.06	+0.11
GPRGNN(O)	70.02±0.7	79.24±1.1	82.24±0.86	84.09±0.81	92.43±0.24	92.99±0.22	95.28±0.04
GPRGNN(M)	70.4±0.41	79.6±0.97	82.19±0.79	84.28±0.86	92.53±0.38	93.33±0.29	95.32±0.15
Δ ↑	+0.38	+0.36	-0.05	+0.19	+0.1	+0.34	+0.04
BernNet(O)	69.12±0.96	78.9±1.04	81.9±0.8	85.15±1.14	92.63±0.29	93.11±0.23	95.3±0.17
BernNet(M)	69.45±0.64	79.07±1.03	82.5±0.78	85.18±0.77	92.58±0.36	93.07±0.29	95.32±0.15
Δ ↑	+0.33	+0.17	+0.6	+0.03	-0.05	-0.04	+0.02

Table 3: Test accuracy of spectral GNNs with/without AdaSpec on homophilic datasets. High accuracy indicates good performance.

From Tables 2 and 3, we observe the following: (1) AdaSpec significantly improves performance on heterophilic graphs compared to homophilic graphs. There is an average accuracy improvement of 1.89% on small heterophilic graphs, an average ROC AUC improvement of 1.27% on large heterophilic graphs, and an average accuracy improvement of 0.43% on homophilic graphs. (2) AdaSpec shows greater performance improvement on small-sized graphs compared to large-sized graphs. The average node classification accuracy improvement on small graphs (Texas, Wisconsin, Cornell) is 3.45%, whereas the improvement on larger graphs (Chameleon, Squirrel) is 0.46%.

The main performance improvement stems from AdaSpec's ability to increase node distinguishability in spectral GNNs. By refining the graph structure representation, AdaSpec enables the model to better separate nodes with similar features or structures. In homophilic graphs, low-frequency components are sufficient for smooth features, so adding more may hurt. Heterophilic graphs require richer spectral patterns, and AdaSpec help by increasing useful frequency components. In small graphs, changes in graph matrix can reveal critical structure. In large graphs, existing structure dominates, changes in graph matrix are less effective.

Component-wise Analysis. We report ChebNet performance augmented with AdaSpec across multiple datasets and conduct an ablation study to isolate the effects of each component. Results in Table 4 show: (1) Full components: Combining all three components consistently yields the best performance. (2) Structure-dominated graphs (e.g., Chameleon, Cora): Ω_D outperforms Ω_S. (3) Feature-dominated graphs (e.g., Texas, Roman_Empire): Ω_S outperforms Ω_D. (4) Frequency components: Increasing non-zero frequency components via $\Omega_F(X)$ improves performance, even when used alone. Each component within AdaSpec independently improves node distinguishability. When combined, these mechanisms complement each other, leading to the strongest overall performance.

AdaSpec	Texas	Chameleon	Roman Empire	Amazon Ratings	Citeseer	Cora
$\Omega_D(A)$	40.75	26.71	22.70	40.75	68.27	81.53
$\Omega_S(A)$	44.51	23.27	54.04	35.28	52.29	55.63
$\Omega_F(X)$	26.24	28.22	54.12	37.16	29.49	65.49
$\Omega(A, X)$	51.16	29.73	54.55	40.92	68.52	82.26

Table 4: Test accuracy of ChebNet with different components of AdaSpec across datasets that $\Omega(A, X)$ contains all three components.

Increased Distinct Eigenvalue Number. We compare the number of distinct eigenvalues between the original normalized adjacency matrix $\tilde{A}$ and the modified matrix $\Omega_D(A)$ from AdaSpec when using ChebNet. Due to the computational cost of full eigendecomposition, we conduct this analysis on small-scale homophilic and heterophilic datasets. As shown in Table 5, $\Omega_D(A)$ consistently increases the number of distinct eigenvalues, supporting Theorem 5.1. Standard normalized adjacency matrix $\tilde{A}$ and its self-loop version $\hat{A}$ are specific cases of the component $\Omega_D(A)$ in AdaSpec by setting $B = 0$ and $B = 1$ respectively. We introduces richer structural information in spectral GNNs by making B learnable matrix (updated via gradient descent) in AdaSpec. The increased number of distinct eigenvalues directly enhances the model's ability to differentiate non-isomorphic nodes.

Dataset	Texas	Wisconsin	Chameleon	Squirrel	Cornell	Citeseer	Cora
$\lvert\mathcal{V}\rvert$	183	251	890	2,223	183	3,327	2,708
$d_{\tilde{A}}$	113	178	845	2,213	122	2,508	2,395
$d_{\Omega_D(A)}$	181	229	888	2,221	144	3,227	2,645
$\triangle \uparrow$	68	51	43	8	22	719	250

Table 5: Number of distinct eigenvalues of the graph matrix. $|\mathcal{V}|$ denotes the number of nodes in graphs. $d_{\tilde{A}}$ and $d_{\Omega_D(A)}$ are numbers of distinct eigenvalues of $\tilde{A}$ and $\Omega_D(A)$ in AdaSpec respectively.

Time Complexity of AdaSpec. We evaluate the training efficiency of ChebNet with and without AdaSpec across multiple datasets. For each dataset, we conduct ten independent runs. We report the average training time per run and the pre-computing time of $\Psi^+(A, X)$ in Table 6. The results show that AdaSpec introduces minimal overhead and can even accelerate convergence on large heterophilic graphs (e.g., Roman_Empire, Amazon_Ratings). When increase graph size from Amazon_Ratings to Coauthor-Physics, the pre-computation time rises from 0.03s to 12.44s, which is consistent with our time complexity analysis in Section 5.4. By incorporating structural and feature bias into the node representation, AdaSpec enables faster convergence and more efficient training.

Datasets	Roman _Empire	Amazon _Ratings	Tolokers	Minesweeper	Questions	Computers	Photo	Coauthor -CS	Coauthor -Physics
ChebNet (O)	1.93	1.91	1.76	1.28	2.53	4.73	3.4	3.67	4.54
ChebNet (S)	1.88	1.35	2.51	2.18	3.05	5.32	4.83	4.11	4.60
$\Delta \uparrow$	-0.05	-0.56	0.75	0.9	0.52	0.59	1.43	0.44	0.06
Pre-Computing	0.26	0.03	0.44	0.08	0.56	1.83	0.9	4.1	12.44

Table 6: Average training and pre-computing time (in seconds) for ChebNet with and without AdaSpec on large heterophilic and homophilic datasets. Pre-computing is for $\Omega_F(X)$ in AdaSpec.

7 Conclusion and Limitations

This work analyzes node distinguishability of spectral GNNs and shows it is governed by the interplay between the graph matrix and node features. Specifically, by the number of distinct eigenvalues and nonzero frequency components in the graph matrix's eigenbasis. We propose AdaSpec, a plug-in module that enhances the node distinguishability of spectral GNNs, offering theoretical guarantees and empirical gains.

While effective, our approach is limited to spectral GNNs and provides only a lower bound on distinguishability. The design of AdaSpec is tailored to certain data distributions and may not generalize universally. Future work could explore more generalizable graph matrix designs, applications to dynamic graphs, and integration with advanced spectral GNNs for broader applicability.

ETHICS STATEMENT

This work presents a theoretical analysis and algorithmic contribution to spectral GNNs for node classification tasks. The research does not involve human subjects, collection of personal data, or direct interaction with individuals. All experiments are conducted on publicly available benchmark datasets that have been widely used in the graph learning community. The proposed AdaSpec is a general-purpose technique for improving node distinguishability in spectral GNNs and does not target specific populations or applications that could raise fairness or discrimination concerns.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we have made every effort to document our methods and experimental setup comprehensively. The main paper provides a complete description of the proposed AdaSpec, including its theoretical derivation and integration with existing GNN architectures. Full proofs for our theoretical claims are provided in Appendix A. All experiments were conducted using publicly available benchmark datasets. Experimental settings, including datasets, preprocessing steps, model architectures, and hyperparameters, are described in detail in the main text and Appendix B. The complete source code, including the implementation of AdaSpec and the scripts to run all experiments, will be released upon acceptance, enabling full reproduction of reported results.

REFERENCES

Adrián Arnaiz-Rodrıéguez, Ahmed Begga, Francisco Escolano, and Nuria M Oliver. Diffwire: Inductive graph rewiring via the lovász bound. In *Learning on Graphs Conference*, pp. 15–1. PMLR, 2022.

Federico Barbero, Ameya Velingker, Amin Saberi, Michael M Bronstein, and Francesco Di Giovanni. Locality-aware graph rewiring in gnns. In *ICLR*, 2024.

Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Francesco Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

Hau Chan and Leman Akoglu. Optimizing network robustness by edge rewiring: a general framework. *Data Mining and Knowledge Discovery*, 30:1395–1425, 2016.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv: Learning*, 2021.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

Yu Tang Guo and Zhewei Wei. Graph neural networks with learnable and optimal polynomial bases. *ArXiv*, abs/2302.12432, 2023. URL https://api.semanticscholar.org/CorpusID:257205644.

David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *ArXiv*, abs/0912.3848, 2009.

Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Mingguo He, Zhewei Wei, and Ji rong Wen. Convolutional neural networks on graphs with chebyshev approximation, revisited. *ArXiv*, abs/2202.03580, 2022a. URL https://api.semanticscholar.org/CorpusID:246652363.

Mingguo He, Zhewei Wei, and Ji rong Wen. Convolutional neural networks on graphs with chebyshev approximation, revisited. *ArXiv*, abs/2202.03580, 2022b. URL https://api.semanticscholar.org/CorpusID:246652363.

Keke Huang, Yu Guang Wang, Ming Li, et al. How universal polynomial bases enhance spectral graph neural networks: Heterophily, over-smoothing, and over-squashing. *arXiv preprint arXiv:2405.12474*, 2024.

Ming Jin, Guangsi Shi, Yuan-Fang Li, Bo Xiong, Tian Zhou, Flora D Salim, Liang Zhao, Lingfei Wu, Qingsong Wen, and Shirui Pan. Towards expressive spectral-temporal graph neural networks for time series forecasting. *IEEE transactions on pattern analysis and machine intelligence*, 2025.

Kedar Karhadkar, Pradeep Kr Banerjee, and Guido Montufar. Fosr: First-order spectral rewiring for addressing oversquashing in gnns. In *ICLR*, 2023.

Thomas Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67:97–109, 2019.

Guoming Li, Jian Yang, and Shangsong Liang. Ergnn: Spectral graph neural network with explicitly-optimized rational graph filters. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Pan Li and Jure Leskovec. The expressive power of graph neural networks. *Graph Neural Networks: Foundations, Frontiers, and Applications*, pp. 63–98, 2022.

Zhengpin Li and Jian Wang. Spectral graph neural networks with generalized laguerre approximation. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7760–7764. IEEE, 2024.

Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *ICLR*, 2023.

Kangkang Lu, Yanhua Yu, Hao Fei, Xuan Li, Zixuan Yang, Zirui Guo, Meiyu Liang, Mengran Yin, and Tat-Seng Chua. Improving expressive power of spectral graph neural networks with eigenvalue correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14158–14166, 2024.

Hongbin Pei, Bingzhen Wei, K. Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *ArXiv*, abs/2002.05287, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023.

Yifang Qin, Wei Ju, Yiyang Gu, Ziyue Qiao, Zhiping Xiao, and Ming Zhang. Polycf: Towards optimal spectral graph filters for collaborative filtering. *ACM Transactions on Information Systems*, 43(4):1–28, 2025.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2020.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *J. Complex Networks*, 9, 2021.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Xu Shen, Pietro Lio, Lintao Yang, Ru Yuan, Yuyang Zhang, and Chengbin Peng. Graph rewiring and preprocessing for graph neural networks based on effective resistance. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

GW Stewart. Matrix perturbation theory. *Computer Science and Scientific Computing/Academic Press, Inc*, 1990.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. 2022.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. *ArXiv*, abs/2205.11172, 2022.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

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

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. Graphsaint: Graph sampling based inductive learning method. *ArXiv*, abs/1907.04931, 2020.

Bingxu Zhang, Changjun Fan, Shixuan Liu, Kuihua Huang, Xiang Zhao, Jincai Huang, and Zhong Liu. The expressive power of graph neural networks: A survey. *arXiv preprint arXiv:2308.08235*, 2023.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, 2020.