ABSTRACT

Graph Neural Networks (GNNs) as spectral graph filters, enhancing specific frequencies of graph signals while suppressing the rest, find a wide range of applications in web networks. To bypass eigendecomposition, polynomial graph filters are proposed to approximate graph filters by leveraging various polynomial bases for filter training. However, no existing studies have explored the diverse polynomial graph filters from a unified perspective for optimization.

In this paper, we first unify polynomial graph filters, as well as the optimal filters of identical degrees into the Krylov subspace of the same order, thus providing equivalent expressive power theoretically. Next, we investigate the asymptotic convergence property of polynomials from the unified Krylov subspace perspective, revealing their limited adaptability in graphs with varying heterophily degrees. Inspired by those facts, we design a novel adaptive Krylov subspace approach to optimize polynomial bases with provable controllability over the graph spectrum so as to adapt various heterophy graphs. Subsequently, we propose AdaptKry, an optimized polynomial graph filter utilizing bases from the adaptive Krylov subspaces. Meanwhile, in light of the diverse spectral properties of complex graphs comprising numerous components, we extend AdaptKry by leveraging multiple adaptive Krylov bases without incurring extra training costs. As a consequence, extended AdaptKry is able to capture the intricate characteristics of graphs and provide insights into their inherent complexity. We conduct extensive experiments across a series of real-world datasets. The experimental results demonstrate the superior filtering capability of AdaptKry, as well as the optimized efficacy of the adaptive Krylov basis.

KEYWORDS

Spectral Graph Neural Networks, Supervised Classification, Krylov Subspace

1 INTRODUCTION

Graph Neural Networks (GNNs) [23], inherently acting as spectral graph filters [5, 10, 23, 42], are widely employed across various web applications, such as classification on web services [16, 51], online e-commerce [11, 48, 50], and analysis of social networks [27, 35, 37]. Specifically, given a graph G, the Laplacian matrix \( L \) is eigendecomposed as \( L = U \Lambda U^T \) where \( U \) is the matrix of eigenvectors and \( \Lambda \) is the diagonal matrix of eigenvalues. Given a signal \( x \), a spectral graph filter \( g_w \) parameterized by vector \( w \) on \( x \) is \( g_w(L)x = Ug_w(\Lambda)U^Tx \). To avert the computation burden of eigendecomposition, polynomial filters have been proposed to approximate the optimal graph filter via polynomial approximation, such as ChebNet [10], BernNet [21], and JacobiConv [44]. Specifically, ChebNet utilizes truncated Chebyshev polynomials [19, 30] and achieves localized spectral filtering. To acquire better controllability and interpretability, BernNet employs Bernstein polynomials [12]. Lately, Wang and Zhang [44] analyze the expressive power of current polynomial filters and propose JacobiConv by exploiting Jacobi polynomial bases [1].

Numerous studies have investigated various polynomials for polynomial filters, yet no prior research has explored the properties of polynomials as signal bases for filter training from a unified perspective. Meanwhile, those existing polynomials are usually exploited directly in graph filters without any customization for different graphs. An intriguing and valuable question that arises is: can we optimize polynomial graph filters by enhancing the underlying polynomial basis?

To address this question, we revisit polynomial filters in terms of Krylov subspace [29, 32]. The Krylov subspace method was initially proposed in the early 1950s for solving linear systems [36, 43]. A Krylov subspace is a subspace of Euclidean vector space. In particular, an order-\( K \) Krylov subspace is constructed by multiplying the first \( K \) powers of a matrix \( A \in \mathbb{R}^{n \times n} \) to a vector \( v \in \mathbb{R}^n \) where \( K \) is an integer (details in Section 6). In the realm of polynomial filters, matrix \( A \) corresponds to the propagation matrix, \( v \) is the feature signal, and \( K \) stands for the propagation hops. In this paper, we demonstrate that polynomial filters sharing identical degrees inherently belong to the Krylov subspace of the same order and hence yield theoretically equivalent expressive power.

From the perspective of the Krylov subspace, we prove that propagation matrices in existing polynomials gradually approach stability in an asymptotic manner regardless of the heterophily in the underlying graphs (Theorem 1). It is known that high-frequency signals can also provide insightful information [2, 7, 25], especially for strong heterophily graphs. Inspired, we design a tunable propagation matrix and propose a novel adaptive Krylov subspace approach. To clarify, we introduce a hyperparameter into propagation matrices by inspecting the graph heat equation [9, 45], equipping the ability to reshape the underlying spectrum of graphs (Theorem 2). As a consequence, polynomials from the adaptive Krylov subspace disrupt the convergence pattern and offer flexible adaptability to graphs with varying heterophily. Subsequently, we propose AdaptKry, an optimized polynomial graph filter by leveraging bases
from the adaptive Krylov subspaces, known as adaptive Krylov bases. Meanwhile, complex graphs consisting of multiple components are inclined to exhibit diverse spectral properties. To handle such scenarios, we extend AdaptKry by employing several adaptive Krylov bases simultaneously. The resultant basis set also sheds light on the complexity of networks. In addition, we integrate weight parameters of different bases into one singular parameter, ensuring that no extra training overhead is incurred. We compare AdaptKry against 11 baselines on 6 real-world datasets with a range of homophily ratios. AdaptKry achieves the highest accuracy scores in node classification for almost all cases. These experimental results strongly support the superior performance of AdaptKry as polynomial filters. We also devise a comprehensive ablation study to demonstrate the properties of adaptive Krylov bases and AdaptKry in Section 5.3. In a nutshell, our contributions are briefly summarized as follows.

- We unify polynomial filters as well as optimal filters from the Krylov subspace perspective and reveal their constrained adaptability in graphs with varying heterophily degrees.
- We devise a novel adaptive Krylov subspace approach and propose an optimized polynomial filter AdaptKry. Meanwhile, we extend AdaptKry to incorporate multiple adaptive Krylov bases, enhancing expressive capability without extra training costs.
- We conduct comprehensive experiments to verify the superior performance of AdaptKry as polynomial filters, as well as extensive ablation study to explore the properties AdaptKry and the adaptive Krylov basis.

2 PRELIMINARY

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G = (\mathcal{V}, \mathcal{E})$</td>
<td>a social network with node set $\mathcal{V}$ and edge set $\mathcal{E}$</td>
</tr>
<tr>
<td>$n, m$</td>
<td>the numbers of nodes and edges in $G$ respectively</td>
</tr>
<tr>
<td>$d_u$</td>
<td>the degree of node $u$</td>
</tr>
<tr>
<td>$N_u, N_u^{(t)}$</td>
<td>the one-hop and $t$-th hop neighbors of node $u$</td>
</tr>
<tr>
<td>$A, D$</td>
<td>the adjacency matrix and diagonal degree matrix</td>
</tr>
<tr>
<td>$\mathcal{L}$</td>
<td>the normalized Laplacian matrix</td>
</tr>
<tr>
<td>$X, z_u$</td>
<td>the feature matrix and the feature vector of node $u$</td>
</tr>
<tr>
<td>$Z, z_u$</td>
<td>the representation matrix and the representation vector of node $u$</td>
</tr>
<tr>
<td>$\mathcal{K}_K(P, x)$</td>
<td>A Krylov subspace constructed by matrix $P$ and vector $x$ as $\mathcal{K}_K(P, x) = \text{span}{x, P_1x, P_2x, \ldots, P^{K-1}x}$</td>
</tr>
</tbody>
</table>

2.1 Notations and Definitions

In this paper, we use bold uppercase letters, bold lowercase letters, and calligraphic fonts to represent matrices (e.g., $A$), vectors (e.g., $x$), and sets (e.g., $N$), respectively. The $i$-th row (resp. column) of matrix $A$ is represented by $A[i, \cdot]$ (resp. $A[\cdot, i]$). We denote $[n] = \{1, 2, \ldots, n\}$.

Figure 1: Illustrations of polynomial approximations to four common graph filters

Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected and connected graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$, where $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$ be the number of nodes and edges, respectively. Let $X \in \mathbb{R}^{n \times d}$ be its $d$-dimensional feature matrix where each node $v \in \mathcal{V}$ is associated with a $d$-dimensional feature vector $x_v \in \mathbb{R}^d$. For simplicity purposes, we also use node $u \in \mathcal{V}$ to indicate its index, i.e., $x_u = X[u, \cdot]$. The direct neighbor set of any node $u \in \mathcal{V}$ with degree $d_u$ is denoted as $\mathcal{N}_u$ with $d_u = |\mathcal{N}_u|$. Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix of $G$, i.e., $A[u, v] = 1$ if $(u, v) \in \mathcal{E}$; otherwise $A[u, v] = 0$, and $D \in \mathbb{R}^{n \times n}$ be the diagonal degree matrix of $G$, i.e., $D[u, u] = d_u$. Normalized Laplacian matrix $\mathcal{L}$ of graph $G$ is defined as $\mathcal{L} = I - \frac{1}{d}AD\frac{1}{d}$ where $I$ is the identity matrix.

2.2 Polynomial Graph Filters

Given a graph $G = (\mathcal{V}, \mathcal{E})$, the corresponding Laplacian matrix is $\mathcal{L} = AUU^\top$, where $U$ is the matrix of eigenvectors and $A = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues. In particular, eigenvalues $\lambda_i \in [0, 2]$ for $i \in [n]$ mark the frequency and the eigenvalue set $\{\lambda_1, \ldots, \lambda_n\}$ is the spectrum of $G$. Without loss of generality, we assume $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \leq 2$.

Given a graph signal $x \in \mathbb{R}^n$, graph Fourier operator $F(x) = U^\top x$ transforms the graph signal $x$ into the spectral domain, and then a predefined spectral filtering $g_w(.)$ parameterized by $w \in \mathbb{R}^n$ is applied on the transformed signals. Last, the processed signal is transformed back via the inverse graph Fourier transform operator $F^{-1}(x) = Ux$. Specifically, the spectral graph filter is defined as

$$F^{-1}(F(g_w \circ F(x)) = U g_w(A) U^\top x$$

$$= U \text{diag}(g_w(\lambda_1), \ldots, g_w(\lambda_n)) U^\top x$$

where $\circ$ is the Hadamard product.

In general, the filter function $g_w(\cdot)$ enhances signals in specific frequency intervals and suppresses the rest parts. Four types of

\footnote{Without further specification, $\mathcal{L}$ refers to the normalized Laplacian matrix in this paper.}
spectral filters are commonly observed in real-world graphs \cite{21}, i.e., low-pass filter, band-pass filter, comb-pass filter, and high-pass filter, as illustrated in Figure 1. Intuitively, homophily graphs tend to contain low-frequency signals whilst heterophily graphs likely own high-frequency signals. To quantify the graph homophily degree, we define homophily ratio \( h \) as follows.

**Definition 1 (Homophily Ratio \( h \)).** Given a graph \( G = (V, E) \) and its label set \( \mathcal{C} \subseteq \mathbb{R}^{|V|} \), the homophily ration \( h \) of \( G \) is the fraction of edges with two end nodes from the same class, i.e.,

\[
h = \frac{|\{(u, v) \in E : C_u = C_v\}|}{|E|}.
\]

Since (i) it is inefficient to conduct matrix eigendecomposition (time complexity \( O(n^3) \)) and (ii) any desired spectral filters can be approximated with polynomials \cite{19, 30}, existing spectral GNNs approximate the spectral filter with various learnable polynomials.

**ChebNet.** Defferrard et al. \cite{10} adopt Chebyshev polynomials up to \( K \)-th order to approximate Equation (1) as

\[
U_{Gw}(\Lambda^t)x \approx \sum_{k=0}^{K} w_k T_k(\hat{\Lambda}) \hat{\Lambda}^t x = \sum_{k=0}^{K} w_k T_k(\hat{L}) x,
\]

where \( \hat{L} = U\hat{A}U^T \) and \( \hat{A} = 2A/\lambda_{\max} - I \) is a normalized diagonal matrix with eigenvalues within \([-1, 1]\), and \( \lambda_{\max} = \max(\lambda_1, \ldots, \lambda_n) \) is the maximum eigenvalue of \( L \). \( w_k \) is the learnable Chebyshev coefficient and \( T_k(x) \) is the Chebyshev polynomial recursively computed as \( T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \) with \( T_0(x) = 1 \) and \( T_1(x) = x \). By taking the learnable Chebyshev coefficient \( w_k \) as the trainable parameter of the \( k \)-th convolution layer, the filter function of ChebNet is \( g_w(\lambda) = \sum_{k=0}^{K} w_k T_k(\lambda) \).

**GPR-GNN.** Chen et al. \cite{8} follow the generalized PageRank (GPR) architecture by stacking up \( K \) convolution layers and assigning a learnable weight for each layer. They propose GPR-GNN as

\[
z = \sum_{k=0}^{K} w_k (D^{-\frac{1}{2}}AD^{-\frac{1}{2}})^k \cdot f_0(x),
\]

where \( \hat{A} = A + I, \hat{D} = D + I \), \( \hat{z} \) is the final node representation, \( f_0(\cdot) \) represents a neural network parameterized by \( \theta \in \mathbb{R}^l \), and \( w_k \) is the learnable parameter of \( k \)-th convolutional layer. Hence, the filter function of GPR-GNN is \( g_w(\lambda) = \sum_{k=0}^{K} w_k (1 - \hat{A})^k \).

**BernNet.** He et al. \cite{21} exploit Bernstein basis \( \left\{ b_k(\cdot) \right\} \) \( 1 \)-\( K \)-order polynomial of the \( k \)-th layer. By piling up \( K \) such convolution layers with trainable parameter \( w_k \), they propose BernNet as

\[
z = \sum_{k=0}^{K} \frac{w_k}{K} \left( 2I - L \right)^{K-k} L^k \cdot f_0(x),
\]

Intuitively, the filter function of BernNet is \( g_w(\lambda) = \sum_{k=0}^{K} \frac{w_k}{K} (1 - \lambda)^{K-k} \).

**JacobiConv.** Recently, Wang and Zhang \cite{44} leverage Jacobi polynomial \cite{1} bases \( P_k(x) \), a general orthogonal polynomials. They devise JacobiConv using Jacobi polynomials as

\[
z = \sum_{k=0}^{K} w_k p^{a,b}_k (1 - L) \cdot x,
\]

where \( w_k \) is the trainable parameter. The corresponding filter function is \( g_w(\lambda) = \sum_{k=0}^{K} w_k p^{a,b}_k (1 - \lambda) \).

### 3 POLYNOMIAL FILTERS VISIT

#### 3.1 Polynomial Filters in Krylov Subspace

The four polynomials introduced in Section 2.2 are essentially four different linearly independent bases of polynomial vector spaces. By simply reordering the terms of the four filter functions, they can be reformulated equally by the monomial sequence \( \{1, x, x^2, \ldots, x^K, \ldots\} \), i.e.,

\[
z = \sum_{k=0}^{K} w_k p_k \cdot x
\]

where \( K \) is the predefined degree of the polynomial with \( K \leq n \), \( w_k \) is the \( k \)-th learnable parameter with \( w \in \mathbb{R}^{K+1} \), and \( p \) is the corresponding propagation matrix. From the perspective of polynomial vector space, the learned polynomial filter \( \sum_{k=0}^{K} w_k p_k \) is the combination of \( K+1 \) polynomial bases of the space with coefficients \( w_k \) for \( k \in \{0, 1, \ldots, K\} \). However, if regarding the product \( p_k^t x \) as the \( k \)-th vector basis, \( p_k^t x \) for \( k \in \{0, 1, \ldots, K\} \) forms the order-\( (K+1) \) Krylov Subspace \( \mathcal{K}_{K+1}(P, x) = \text{span}(x, p_0^t x, \ldots, p_K^t x) \), and the filtered signal \( z = \sum_{k=0}^{K} w_k p_k^t x \) is deemed as the weighted combination of the bases of \( \mathcal{K}_{K+1}(P, x) \). As such, we establish the following proposition.

**Proposition 1.** Given a graph \( G \) and a graph signal \( x \), the set of all \( K \)-order polynomial filters with propagation matrix \( P \) belongs to the Krylov subspace \( \mathcal{K}_{K+1}(P, x) \).

The aim of polynomial filters is to approximate the spectral filter \( U_{Gw}(\Lambda)^t x \) so as to avoid eigendecomposition. In what follows, we prove that the spectral filter can be equivalently expressed by polynomial filters constructed from a Krylov subspace with order \( K \) determined by both matrix \( P \) and vector \( x \). Before proceeding further, we first introduce the concept of vector grade with respect to a matrix.

**Definition 2 (Grade of \( x \) with respect to \( P \) \cite{29}).** The grade of \( x \) with respect to \( P \) is a positive integer \( t \) such that

\[
\dim \mathcal{K}_K(P, x) = \min(K, t).
\]

The grade \( t \) of \( x \) with respect to \( P \) defines the highest dimension of a Krylov subspace constructed by \( P \) and \( x \). Ergo, the new vectors \( p_k^t x \) are linearly dependent on the previous ones and \( \mathcal{K}_K(P, x) = \mathcal{K}_{K+1}(P, x) \) for \( K \geq t \). As a consequence, we have the following conclusion relating to the optimal spectral filter.

**Proposition 2.** Consider a propagation matrix \( P = I - L \) and a graph signal \( x \) with grade \( t \) with respect to \( P \). There exists a \( (t - 1) \)-order polynomial filter \( \sum_{k=0}^{t-1} p_k^t x \) parameterized by \( \theta \in \mathbb{R}^l \) equivalent to the optimal spectral filter \( U_{Gw}(\Lambda)^t x \).

Proposition 2 implies that the utilization of \( K \)-order polynomial filters approximates the spectral filter by preserving only the first \( K \) bases from the order-\( t \) Krylov subspace \( \mathcal{K}_t(P, x) \) and discarding the rest bases.
3.2 Theoretical Analysis on Bases Convergence
As introduced in Section 2.2, polynomial filters utilize different propagation matrices $P$. To elaborate, ChebNet sets $P = 2L/\lambda_{\text{max}} - I$ where $\lambda_{\text{max}}$ is the largest eigenvalue of $L$. GPR-GNN exploits $P = I - \hat{L}$ where $\hat{L} = \frac{1}{2}\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$. BernNet employs $P = I - \hat{L}/2$ while JacobiiConv adopts $P = I - \hat{L}$. In general, propagation matrix $P$ is derived from the Laplacian matrix $L$ or its variant $\hat{L}$. Typically, eigenvalues of $P$ are bounded in the interval $(−1, 1)^2$.

Towards polynomial bases, one pertinent and crucial problem is the convergence determined by propagation matrix $P$, which in turn influences the selections of polynomial order $K$. To this end, we first prove that for a sufficiently large $K$, the basis vector $P^k x$ converges to $\phi \tilde{\phi}^2 x$ where $\phi$ is the eigenvector associated with the largest eigenvalue of $P$. Consequently, we establish an upper bound for the choice of an appropriate $K$ in terms of $P$ measured by the relative pointwise distance [39].

Theorem 1. Let $\lambda_i(P)$ be the $i$-th eigenvalue of the matrix $P$ where $i \in \{1, 2, \ldots, n\}$ and $\lambda^* = \max(-\lambda_1(P), \lambda_{n-1}(P))$. It holds that (i) $P^k x$ converges asymptotically to a stable propagation matrix $P_\epsilon$, i.e., $P_\epsilon := \lim_{K \to \infty} P^k$, and (ii) given $\epsilon \in (0, 1)$, $\max_{u \in \mathbb{V}} |P^k u, v] - P_\epsilon u, v] | \leq \epsilon P_\epsilon u, v]$ if $K \geq \ln \frac{\epsilon m}{\ln \lambda^*}$ where $d_{\text{min}} := \min_{u \in \mathbb{V}} d_u$.

Theorem 1 elucidates that the convergence speed of the polynomial basis is dominated by value $\lambda^*$, i.e., the second largest absolute value among eigenvalues of the propagation matrix $P$. In particular, a smaller $\lambda^*$ corresponds to a more rapid convergence speed. Moreover, the empirical efficacy of polynomial filters is substantially determined by $P$ which plays a pivotal role in the construction of Krylov subspace. Yet, propagation matrices in existing polynomials asymptotically converge towards stability regardless of the heterophily degrees of underlying graphs. As a consequence, a crucial question is how we can optimize the design of the propagation matrix to enhance the expressive power of polynomial filters. We will answer this question in Section 4.

3.3 Connection to Spatial GNNs
Polynomial graph filters approximate the underlying spectral graph filters by leveraging different polynomials. By identifying the propagation matrices $P$, they can be expressed equivalently as $\sum_{k=0}^{K-1} w_k P^k x$. The weight parameter $w_k$ inherently stems from the coefficients of original polynomials and learnable weights. From a spatial perspective, the order $K$ of propagation matrix $P$ is the propagation step in feature aggregation from neighbors. Therefore, the fundamental distinction among diverse polynomials lies in that they assign varying weights to neighbors across multiple hops. Meanwhile, as analyzed in Section 3.2, different propagation matrices $P$ exhibit divergent convergence properties. Those disparities lead to distinct empirical performance, attributed to the different capabilities of the polynomials to capture the intricate characteristics of underlying graphs.

Proposition 2 reveals that propagation within the grade $t$ hops of signal $x$ with respect to propagation matrix $P$ suffices. This fact hints that $\forall v \in \mathbb{V}$ receives all signals from neighbors within $t$ hops $t = 1$ is achievable when graphs are bipartite, which is not considered in our paper.

3.4 POLYNOMIAL FILTER FROM ADAPTIVE KRYLOV SUBSPACE
4.1 Adaptive Krylov Subspace
As proven, existing polynomial filters belong to the Krylov subspace $\mathcal{K}(P, x)$ and have theoretically equivalent expressive power, wherein the propagation matrix $P$ plays a crucial role. Despite that, matrix $P$ in existing filters is predefined and fixed regardless of the homorophy ratios of underlying graphs. Yet, responded frequencies are significantly correlated with homorophy ratios.

For ease of exposition, we consider a binary node classification for a graph $G = (\mathcal{V}, \mathcal{E})$ in homorophy ratio $h$ associated with a node signal $x \in \mathbb{R}^n$. Let $y_0 \in \mathbb{R}^n$ (resp. $y_1 \in \mathbb{R}^n$) be the one-hot label vector such that $y_{0, i} = 1$ (resp. $y_{1, i} = 1$) if node $u_i$ belongs to class 0 (resp. class 1), otherwise $y_{0, i} = 0$ (resp. $y_{1, i} = 0$). Let $y = y_0 - y_1$ be the label differences. Therefore, we can establish that

$$y^\top L x = (D^{-\frac{1}{2}} y^\top (1 - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) D^{-\frac{1}{2}} y) = y^\top D^{-\frac{1}{2}} (1 - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) D^{-\frac{1}{2}} y$$

$$= \frac{1}{m} \sum_{i=1}^{m} d_i y_i^2 = \frac{1}{m} \sum_{i=1}^{m} d_i y_i^2 = 4(1 - h) m = 4(1 - h).$$

Since $y^\top y = n$, we have $4(1 - h)n = y^\top L y$. Meanwhile, let $L = UAU^\top$ where $U$ is the eigenvector matrix and $\Lambda$ is the diagonal matrix of eigenvalue spectrum $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. W.l.o.g., suppose $U^\top y = (a_1, a_2, \ldots, a_n)$ where $a_i$ is the projection (response) of $y$ on eigenvector $U^\top[i, :]$. Then we have $y^\top L y = y^\top UAU^\top y = \sum_{i=1}^{n} a_i^2 \lambda_i$. Therefore, we have $\frac{1}{n} \sum_{i=1}^{n} a_i^2 \lambda_i = 4(1 - h)$, i.e., graphs with different homorophy ratios respond to different frequencies.

This fact motivates us to design an adaptive $P$ that provides better adaptability and controllability. Recall that the propagation matrix $P$ is primarily derived from the Laplacian matrix $L$ and explored in the graph heat diffusion. Hence, we resort to the following Graph Heat Equation [9, 45].

$$\frac{dH_t}{dt} = -LH_t, \quad H_0 = X. \quad (7)$$

where $H_t$ is the node representations of graph $G$ at time $t$. Given an infinitesimal interval $\tau$, the Euler method implies that

$$H_{t+\tau} = \lim_{\tau \to 0^+} H_t - \tau L H_t$$

$$\lim_{\tau \to 0^+} H_t = H_t(1 - \tau L)$$

$$\lim_{\tau \to 0^+} H_t((1 - \tau)L) + \tau D^{-\frac{1}{2}} A D^{-\frac{1}{2}}. \quad (8)$$

Equation (8) demonstrates that $(1 - \tau)L + \tau D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ propagates $H_t$ forward with a stride $\tau$. In light of this, we design our propagation matrix $P_t$ via renormalization trick [23] as

$$P_t = (1 - \tau)L + \tau D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \frac{1}{\tau}.$$

where $A_t = \tau A + (1 - \tau)I$ and $D_t = \tau D + (1 - \tau)$I. In particular, $\tau$ decides the percentages of signals from ego parts and neighbors in feature aggregations. In the meantime, the $\tau$ value is able to alter the graph spectrum as follows.

4.2 Experimental Results

In this section, we present a comprehensive analysis of our proposed polynomial filters through a series of experiments.
\[ L_\tau = I - P_\tau \text{ and } \lambda_i(\tau)_{i=1}^\tau \text{ denote the eigenvalues of } L_\tau \text{ with } \lambda_1(\tau) \leq \lambda_2(\tau) \leq \cdots \leq \lambda_n(\tau) \text{ for } \tau > 0. \] Then it holds that \( \lambda_i(\tau) \) is a monotonically increasing function of \( \tau \) and \( \lambda_i(\tau) \leq \lambda_i \) if \( \tau \in (0, 1] \) and \( \lambda_i(\tau) > \lambda_i \) if \( \tau > 1 \) for \( i \in [n] \).

Theorem 2 reveals that changing \( \tau \) is capable of reshaping the underlying spectrum of \( L_\tau \). In doing so, we are able to build adaptive Krylov subspace \( \mathcal{K}_K(P_\tau, x) \) such that the corresponding Krylov basis, known as adaptive Krylov basis, has the adaptability to graphs with varying homophily ratios.

### 4.2 Polynomial Filter AdaptKry

In this section, we propose AdaptKry, an optimized polynomial filter utilizing the adaptive Krylov basis from the adaptive Krylov subspace \( \mathcal{K}_K(P_\tau, x) \). Specifically, AdaptKry first initializes a weight vector \( w \in \mathbb{R}^{k+1} \) and then constructs the Krylov basis \( \{x, P_\tau x, \cdots, P_k^\tau x\} \). Next, those vectors are concatenated and multiplied by \( w \) in an element-wise manner, after which it is fed into an arbitrary Multilayer Perceptron (MLP) network to train the weight parameter \( w \). The pseudo-code for AdaptKry is illustrated in Algorithm 1. It is worth mentioning that AdaptKry, in contrast to existing polynomial filters, separates the procedures of feature propagation and weight learning. This decoupling mechanism enables adaptive Krylov bases unchanged during training and thus averts the repetitive calculation on polynomials.

**Algorithm 1: AdaptKry**

**Input:** Graph \( G \), feature matrix \( X \), order \( K \), step size \( \tau \)

**Output:** \( Z \)

1. \( Z \leftarrow X \cdot F(0) = X \)
2. Weight vector \( w \) initialization;
3. \( A_\tau = rA + (1 - r)I, D_\tau = rD + (1 - r)I, P_\tau = D_\tau^{-\frac{1}{2}}A_\tau D_\tau^{-\frac{1}{2}} \)
4. for \( \ell = 1 \) to \( K \) do
5. \( F(\ell) \leftarrow P_\tau^{-\ell} F(\ell-1) \)
6. \( Z \leftarrow \text{concatenate}(Z, F(\ell)) \)
7. \( Z \leftarrow \text{MLP}(Z \odot w) \)
8. return \( Z \)

### 4.3 Approximation Analysis

Theoretically, AdaptKry provides equivalent expressive power with existing polynomial graph filters. However, by equipping adaptability to AdaptKry, it provides the ability to alter the underlying spectrum as desired. In this regard, AdaptKry derived from adaptive Krylov subspace yields an improved capability to capture the spectral characteristics of underlying graphs, compared to existing polynomial filters from ordinary Krylov subspace. Furthermore, we analyze in Section 3.2 that the convergence rate of polynomials is governed by value \( \lambda^* \) determined by eigenvalues of propagation matrices. Therefore, it is intuitive that reshaping the underlying spectrum enhances the expressive capabilities of polynomials.

**Theorem 3.** Consider a connected graph \( G = (V, E) \) with graph signal \( x \in \mathbb{R}^n \). Let \( F_1(x) = \sum_{t=0}^{l-1} w_t^0 P_t^k x \) and \( F_2(x) = \sum_{t=0}^{l-1} w_t^1 P_t^k x \) are the optimal polynomial filters from ordinary Krylov subspace \( \mathcal{K}_K(P, x) \) and adaptive Krylov subspace \( \mathcal{K}_K(P_\tau, x) \) respectively where \( t \) is the grade of graph signal to the propagation matrix \( \mathbb{Z} \). When training models from the two Krylov subspaces, we assume the initial weight \( w \) is randomly sampled from a predefined distribution. It holds that \( \mathbb{E} ||\omega^d - \omega||_2 \leq \mathbb{E} ||\omega^a - \omega||_2 \) where the expectation is over the randomness of \( w \).

### Discussion on basis orthogonality

Theoretically, distinct polynomial bases yield equivalent expressive capabilities, as demonstrated in Proposition 1. The major distinction lies in the convergence rates of gradient descent in the process of model training. The convergence rate reaches optimal if the condition number of the Hessian matrix of gradients is minimal, as demonstrated in [4, 46]. As for polynomial filters, the condition number becomes minimum when polynomials form an orthonormal basis with respect to the underlying weight function [44] which, however, is intractable. Instead, as stated in [17, 29], such orthonormal basis can be directly established by orthogonalizing the basis of \( \mathcal{K}_K(P, x) \) via three-term recurrence method [15, 29] without resolving the weight function. Despite that, there are two potential weaknesses. First, similar to existing polynomials, orthonormal bases demonstrate constrained flexibility in accommodation to graphs with diverse heterophily, thus yielding suboptimal empirical performance, as confirmed in our ablation study in Section 5.3. Second, the orthogonalization of the basis incurs extra \( O(Kn) \) computation costs.

\[ \frac{\text{Algorithm 1: AdaptKry}}{\text{Input: Graph } G, \text{ feature matrix } X, \text{ order } K, \text{ step size } \tau}{\text{Output: } Z}
\]

1. \( Z \leftarrow X \cdot F(0) = X \)
2. Weight vector \( w \) initialization;
3. \( A_\tau = rA + (1 - r)I, D_\tau = rD + (1 - r)I, P_\tau = D_\tau^{-\frac{1}{2}}A_\tau D_\tau^{-\frac{1}{2}} \)
4. for \( \ell = 1 \) to \( K \) do
5. \( F(\ell) \leftarrow P_\tau^{-\ell} F(\ell-1) \)
6. \( Z \leftarrow \text{concatenate}(Z, F(\ell)) \)
7. \( Z \leftarrow \text{MLP}(Z \odot w) \)
8. return \( Z \)
Table 2: Dataset Details.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
<th>Actor</th>
<th>Chameleon</th>
<th>Squirrel</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Nodes (n)</td>
<td>2,708</td>
<td>3,327</td>
<td>19,717</td>
<td>7,600</td>
<td>2,277</td>
<td>5,201</td>
</tr>
<tr>
<td>#Edges (m)</td>
<td>5,429</td>
<td>4,732</td>
<td>44,338</td>
<td>26,659</td>
<td>31,371</td>
<td>198,353</td>
</tr>
<tr>
<td>#Features (d)</td>
<td>1,433</td>
<td>3,703</td>
<td>500</td>
<td>932</td>
<td>2,325</td>
<td>2,089</td>
</tr>
<tr>
<td>#Classes</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Homo. ratio (h)</td>
<td>0.81</td>
<td>0.73</td>
<td>0.80</td>
<td>0.22</td>
<td>0.23</td>
<td>0.22</td>
</tr>
</tbody>
</table>

5 EXPERIMENTS

In this section, we aim to i) evaluate the capability of AdaptKry as a graph filter for node classification on both homophily and heterophily graphs against a series of baselines and ii) validate the design of AdaptKry and the properties of adaptive Krylov bases. The classification performance is measured in accuracy.

5.1 Experimental Setting

**Datasets.** We evaluate the methods on 6 real-world datasets with varied sizes and homophily ratios, as presented in Table 2. In particular, the 3 citation networks [40], i.e., Cora, Citeseer, and Pubmed, are homophily graphs with homophily ratios 0.81, 0.73, and 0.80 respectively; the Wikipedia graphs Chameleon and Squirrel, the actor co-occurrence graph from WebKB3 [34] are heterophily graphs with homophily ratios 0.22, 0.23, and 0.22 respectively.

By following the convention [20, 21, 44], we i) generate 10 random splits of training/validation/testing with 60%/20%/20% percents respectively for the 6 datasets and report the average accuracy with associated standard deviations for each method.

**Algorithms.** We consider two seminal GNN models GCN [23] and the simplified SGC [47], and also include SIGN [14] and ASGC [6] as they are two variants of SGC for simplicity. Five polynomial graph filters, i.e., GPR-GNN [8], EvenNet [26], ChebNet [10] and its improved version ChebNetII [20], BernNet [21], and JacobianConv [44], and one recently proposed orthogonal basis model OptBasisGNN [17] are deemed as the competitors of polynomial filters. We compare our AdaptKry against 11 baselines on the 6 datasets. Appendix A.2 provides the links to the source codes of baselines.

**Parameter Settings.** There are 4 key hyperparameters in AdaptKry, i.e., the neighbor hop K, the step size τ, learning rate and hidden dimension. For a fair comparison, we fix K = 10 for all tested methods. We tune these parameters to acquire the best attainable performance on each graph according to their homophily ratios, as summarized in Table 6 in Appendix A.2. For baselines, we either adopt the public results if applicable or follow the original settings in [21, 44, 47] for the best possible performance. More details are discussed in Appendix A.2.

5.2 Node Classification

Table 3 presents the accuracy associated with the standard deviation of all tested 12 methods on the 6 datasets with homophily ratios from 0.22 ~ 0.81. For ease of exposition, we highlight the highest accuracy score in bold and underline the second highest accuracy score for each dataset.

First of all, observe that AdaptKry achieves the highest accuracy scores almost on all datasets except a slight lag on Squirrel. In particular, AdaptKry advances the accuracy score up to 3.0% and 1.18% on the homophily datasets Citeseer and Pubmed. These observations strongly support the state-of-the-art capability of AdaptKry as polynomial filters by utilizing the adaptive Krylov basis. Meanwhile, it also confirms our analysis in Section 4.3. As analyzed in Section 4.1, the notable advantage of AdaptKry arises from the adaptive parameter τ. This enables AdaptKry to adapt graphs with a wide range of homophily ratios as various heterophily graphs suit different τ values, further explored in Section 5.3.

5.3 Ablation Study

**Sensitivity on parameter τ.** As stated in Section 4.1, τ in propagation matrix Pτ controls the portions of signals from ego parts and neighbors in the feature aggregation. To evaluate the sensitivity of AdaptKry towards τ for different datasets, we test AdaptKry by setting τ = {0.1, 0.2, 0.3, ..., 0.9, 1.0} on one homophily graph (Cora, homo. ratio 0.81) and one heterophily graph (Squirrel, homo. ratio 0.22). The resultant accuracy scores are plotted in Figure 2. As shown, the accuracy scores on Cora remain close for varying τ, whilst those on Squirrel rise along with the increase of τ and reach the peak at τ = 0.9. This reveals that heterophily graphs are more sensitive to the τ value compared with homophily graphs since feature signals of connected neighbors from heterophily graphs differ more. This result further confirms our claim that graphs with different homophily ratios respond to different frequencies in Section 4.1.

**Spectrum reshaping by τ.** As proved in Theorem 2, the parameter τ within the propagation matrix Pτ is able to reshape the graph spectrum, thereby offering underlying polynomial basis the adaptability. To elaborate the influence of τ on polynomial bases, we generate three sets of order-K polynomial bases with τ = {0.5, 1.0, 1.5} for Cora and Squirrel. The resultant bases Z = Z ∈ R^{n × (K+1)d} for each τ value where d is the feature dimension. Specifically, for feature signal x ∈ R^n from each dimension of feature matrix X ∈ R^{n × d}, we compute the angles between signal vectors in two consecutive hops within the basis, leading to a total of Kd angles. Subsequently, these angles are averaged for the d signals, resulting in K angles that characterize the polynomial basis. In particular, we keep K = 10 for this experiment.

Figure 3 illustrates the evolving patterns of angles between two consecutive signal vectors in polynomial bases across three different τ values. Observe that polynomial bases with τ = 0.5 and τ = 1.0 all converge steadily, which is consistent with Theorem 1. In particular, basis with τ = 0.5 converges faster than that...
Table 3: Accuracy (%) on small datasets.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
<th>Actor</th>
<th>Chameleon</th>
<th>Squirrel</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>87.18 ± 1.12</td>
<td>79.85 ± 0.78</td>
<td>86.79 ± 0.31</td>
<td>33.26 ± 1.15</td>
<td>60.81 ± 2.95</td>
<td>45.87 ± 0.88</td>
</tr>
<tr>
<td>SGC</td>
<td>86.83 ± 1.28</td>
<td>79.65 ± 1.02</td>
<td>87.14 ± 0.90</td>
<td>34.46 ± 0.67</td>
<td>44.81 ± 1.20</td>
<td>25.75 ± 1.07</td>
</tr>
<tr>
<td>ASGC</td>
<td>85.35 ± 0.98</td>
<td>76.52 ± 0.36</td>
<td>84.17 ± 0.24</td>
<td>33.41 ± 0.80</td>
<td>71.38 ± 1.06</td>
<td>57.91 ± 0.89</td>
</tr>
<tr>
<td>SIGN</td>
<td>87.70 ± 0.69</td>
<td>80.14 ± 0.87</td>
<td>89.09 ± 0.43</td>
<td>41.22 ± 0.96</td>
<td>60.92 ± 1.45</td>
<td>45.59 ± 1.40</td>
</tr>
<tr>
<td>ChebNet</td>
<td>87.32 ± 0.92</td>
<td>79.33 ± 0.57</td>
<td>87.82 ± 0.24</td>
<td>37.42 ± 0.58</td>
<td>59.51 ± 1.25</td>
<td>40.81 ± 0.42</td>
</tr>
<tr>
<td>GPR-GNN</td>
<td>88.54 ± 0.67</td>
<td>80.13 ± 0.84</td>
<td>88.46 ± 0.31</td>
<td>39.91 ± 0.62</td>
<td>67.49 ± 1.38</td>
<td>50.43 ± 1.89</td>
</tr>
<tr>
<td>BernNet</td>
<td>88.51 ± 0.92</td>
<td>80.08 ± 0.75</td>
<td>88.51 ± 0.39</td>
<td>41.71 ± 1.12</td>
<td>68.53 ± 1.68</td>
<td>51.39 ± 0.92</td>
</tr>
<tr>
<td>JacobiConv</td>
<td>88.98 ± 0.72</td>
<td>80.78 ± 0.79</td>
<td>89.62 ± 0.41</td>
<td>41.17 ± 0.64</td>
<td>74.20 ± 1.03</td>
<td>57.38 ± 1.25</td>
</tr>
<tr>
<td>EvenNet</td>
<td>87.77 ± 0.67</td>
<td>78.51 ± 0.63</td>
<td>90.87 ± 0.34</td>
<td>40.36 ± 0.65</td>
<td>67.02 ± 1.77</td>
<td>52.71 ± 0.85</td>
</tr>
<tr>
<td>ChebNetII</td>
<td>88.71 ± 0.93</td>
<td>80.53 ± 0.79</td>
<td>88.93 ± 0.29</td>
<td>41.75 ± 1.07</td>
<td>71.37 ± 1.01</td>
<td>57.72 ± 0.59</td>
</tr>
<tr>
<td>OptBasisGNN</td>
<td>87.00 ± 1.55</td>
<td>80.58 ± 0.82</td>
<td>90.30 ± 0.19</td>
<td>42.39 ± 0.52</td>
<td>74.26 ± 0.74</td>
<td>63.62 ± 0.76</td>
</tr>
</tbody>
</table>

Table 4: Accuracy (%) and corresponding $\tau$ set.

<table>
<thead>
<tr>
<th>$\tau$ set</th>
<th>Cora</th>
<th>Actor</th>
<th>Chameleon</th>
<th>Squirrel</th>
</tr>
</thead>
<tbody>
<tr>
<td>${\tau_1}$</td>
<td>88.90 ± 1.70</td>
<td>42.06 ± 1.27</td>
<td>74.09 ± 1.22</td>
<td>63.31 ± 0.76</td>
</tr>
<tr>
<td>${\tau_1, \tau_2}$</td>
<td>89.85 ± 1.36</td>
<td>42.16 ± 1.73</td>
<td>74.53 ± 1.21</td>
<td>63.61 ± 0.76</td>
</tr>
<tr>
<td>${\tau_1, \tau_2, \tau_3}$</td>
<td>89.95 ± 0.95</td>
<td>42.70 ± 1.14</td>
<td>73.83 ± 0.77</td>
<td>63.31 ± 0.76</td>
</tr>
<tr>
<td>${\tau_1}$</td>
<td>[0.8]</td>
<td>[0.7]</td>
<td>[0.8]</td>
<td>[0.8]</td>
</tr>
<tr>
<td>${\tau_1, \tau_2}$</td>
<td>[0.8, 1.0]</td>
<td>[0.6, 1.4]</td>
<td>[0.5, 0.8]</td>
<td>[0.8, 0.8]</td>
</tr>
<tr>
<td>${\tau_1, \tau_2, \tau_3}$</td>
<td>[0.5, 0.8, 1.1]</td>
<td>[0.6, 1.7, 1.8]</td>
<td>[0.5, 0.8, 1.1]</td>
<td>[0.8, 0.8, 0.8]</td>
</tr>
</tbody>
</table>

Table 5: Accuracy (%) of orthogonal / non-orthogonal polynomials

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>hop $K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>OrthKry</td>
<td>87.37</td>
</tr>
<tr>
<td></td>
<td>AdaptKry</td>
<td>87.85</td>
</tr>
<tr>
<td></td>
<td>OrthKry</td>
<td>81.88</td>
</tr>
<tr>
<td></td>
<td>AdaptKry</td>
<td>83.82</td>
</tr>
<tr>
<td></td>
<td>OrthKry</td>
<td>72.14</td>
</tr>
<tr>
<td></td>
<td>AdaptKry</td>
<td>68.84</td>
</tr>
<tr>
<td></td>
<td>OrthKry</td>
<td>62.21</td>
</tr>
<tr>
<td></td>
<td>AdaptKry</td>
<td>60.31</td>
</tr>
</tbody>
</table>

Table 4 presents the obtained accuracy scores and the corresponding $\tau$ set. We highlight the highest score and the associated $\tau$ set in bold. When achieving the highest scores, AdaptKry adopts three polynomial bases on both Cora and Actor and uses two and one bases on Chameleon and Squirrel respectively. The exact number and value of $\tau$ manifest the signal information contained in the datasets. Intuitively, singular set $\{\tau_1\}$ indicates the primary signal frequency, while sets $\{\tau_1, \tau_2\}$ and $\{\tau_1, \tau_2, \tau_3\}$ are able to uncover the signal distribution. Specifically, the fact that $\tau_1 > \tau_2$ shows that high-frequency (larger $\tau$) signal benefits node representations. For the case of dataset Actor (resp. Chameleon), it hints that the high-frequency (resp. low-frequency) signal dominates the others.

Orthogonal bases and computation overheads. We exploit the three-term recurrence method to orthogonalize polynomial bases and implement the orthogonal version OrthKry of AdaptKry. Since OrthKry fixes the relative positions of polynomial bases into orthogonality, it cannot take advantage of the utilization of multiple bases. For a fair comparison, we adopt a singular basis for AdaptKry as well. In particular, we test OrthKry and AdaptKry on two homophily datasets Cora and Citeseer, and two heterophily datasets Chameleon and Squirrel.

Table 5 reports the accuracy scores with propagation hop $K \in \{2, 4, 6, 8, 10\}$.
OrthKry evaluates consistently higher accuracy scores than OrthKry for different propagation hop $K$ values. On the heterophily dataset Chameleon, the performance of OrthKry closely aligns with that of AdaptKry as they outperform each other alternatively. On dataset Squirrel, we observe that OrthKry demonstrates performance gains over AdaptKry. In a nutshell, AdaptKry exhibits clear advantages over OrthKry on homophily datasets while none of them presents dominant filtering capability on heterophily datasets. Therefore, non-orthogonal polynomials in AdaptKry exhibit general and comparable filter capability with orthogonal polynomials.

Figure 4 compares the running times, i.e., feature propagation times and training times of AdaptKry and OrthKry on datasets Citeseer and Squirrel. As displayed, AdaptKry demonstrates notable efficiency advantages over OrthKry. The primary reason is that the adaptive Krylov basis in AdaptKry accommodates the underlying better than the fixed orthogonal basis in OrthKry, as proved in Theorem 3. As we also mentioned in Section 4.3, the orthogonal basis in OrthKry incurs extra $O(Kn)$ computation overheads compared with the non-orthogonal basis.

6 RELATED WORK

Krylov subspace method. The idea of the Krylov subspace method was originally conceived by Lanczos, Hestenes, and Stiefel in the early 1950s for solving linear algebraic systems [18, 32]. An order-$K$ Krylov subspace is constructed by multiplying the first $K$ powers of a $n \times n$ matrix $A$ to an $n$-dimension vector $v$, i.e., $\mathcal{K}_K(A, v) = \text{span}\{v, Av, \cdots, A^{K-1}v\}$. In doing so, the Krylov subspace as a subspace of $\mathbb{R}^n$ can capture the property of $A$ [36]. Specifically, for a large linear system $Ax = v$ to identify the unknown $x \in \mathbb{R}^n$, the Krylov subspace method constructs a subspace $\mathcal{K}_m(A, v)$ with $m \ll n$ and then finds an approximate solution $\hat{x}$ that belongs to $\mathcal{K}_m(A, v)$. Krylov subspace method is widely used in iterative methods in linear algebra [29].

Polynomial graph filters. The aim of research on graph spectral filters is to enhance expressive capabilities through various polynomials to approximate the optimal graph filters. ChebNet [10] utilizes a truncated Chebyshev polynomial [19, 30] up to $K$ orders with each polynomial basis assigned a learnable parameter, which makes ChebNet a general localized graph filters [41]. By simplifying ChebNet, GCN [23] retains the first two convolution layers and takes a renormalized propagation matrix as convolution operations, which makes it a low-pass filter. To better capture node proximity, APPNP [24] adopts Personalized PageRank (PPR) [33, 38] and takes the PPR values of neighbors as fixed aggregation weights, resulting in a low-pass graph filter as well. To achieve better adaptability, GPR-GNN [8] exploits the generalized PageRank (GPR) matrix with trainable weights for convolutional layers. Similarly, GNN-LF/HF proposed by Zhu et al. [53] forms a low-pass and high-pass filter respectively by devising the variants of the Laplacian matrix as the propagation matrices. ARMA [3] resorts to rational convolutional filters using auto-regression moving average filters [31] instead of polynomials, aiming to yield flexible frequency response. However, to approximate the inverse matrix, ARMA inevitably incurs large computation overheads. He et al. [21] employ Bernstein polynomials [12] and propose BernNet for better controllability and interpretability. However, feature propagation in BernNet leads to quadratic time complexity of hop $K$. To improve the flexible controllability on concentration center and bandwidth, Li et al. [28] combines a series of Gaussian bases on approximation and propose $G^2$CN. In the meanwhile, Wang and Zhang [44] investigate the expressive power of polynomial filters and propose JacobicConv by adopting Jacobi polynomial [1] bases. Later, He et al. [20] revisit Chebyshev approximation and point out the over-fitting issue in ChebNet. To resolve the issue, they propose ChebNetII based on Chebyshev interpolation. Recently, Guo and Wei [17] orthogonalize the polynomial bases for optimal convergence and propose OptBasisGNN. ASGC [6] simplifies graph convolution operation inspired by SGC [47] for heterophily graphs. To this end, a Krylov matrix is calculated and equipped with coefficients. However, ASGC presents suboptimal classification performance in the experiments. SIGN is another SGC-based method similar to AdaptKry. However, SIGN filters out the negative values in Krylov basis before weight learning. This operation damages the polynomial properties of the Krylov basis and unable to approximate graph filters. There are also other newly proposed GNN models, e.g., Nodeformer [49], NIGCN [22], and Auto-HeG [52], but they do not exhibit explicit spectral properties and thus are not discussed here.

7 CONCLUSION

In this paper, we unify existing polynomial filters and optimal filters into the Krylov subspace and reveal their limited adaptability for diverse heterophily graphs. To optimize polynomial filters, we design a novel adaptive Krylov subspace with provable controllability over the graph spectrum, enabling adaptability to graphs in varying heterophily degrees. Consequently, we propose AdaptKry by leveraging this adaptive Krylov basis. Furthermore, we extend AdaptKry to incorporate multiple adaptive Krylov bases without incurring extra training costs, thereby accommodating graphs with diverse spectral properties and offering insights into the inherent complexities. The extensive experiments and ablation studies strongly support the superior performance of AdaptKry as polynomial filters, as well as the optimized expressive capability of the adaptive Krylov basis.
A APPENDIX

A.1 Proofs

Proof of Proposition 1. Suppose a $K$-order polynomial graph filter $GF$ with propagation matrix $P_M$. Let $f(x, i) = \sum_{k=0}^{K} \phi_k x^k$ be the $i$-th polynomial basis of $GF$ where $\Phi \in \mathbb{R}^{(K+1) \times (K+1)}$ is the coefficient matrix and $\Phi[i, k] = \phi_k$. Consequently, representation $Z$ from $GF$ is calculated as $Z = \sum_{k=0}^{K} w_k f(P_M, i)X = \sum_{k=0}^{K} w_k \sum_{i=0}^{K} \phi_k p_k i M X$ where $w \in \mathbb{R}^{K+1}$ is the learnable parameter vector. By choosing $\Theta = \{\theta_0, \theta_1, \ldots, \theta_K\} \in \mathbb{R}^{K+1}$ with $\Theta = \Phi^\top w$ as the coefficients of basis $(X, P_M X, P_M^2 X, \ldots, P_M^K X)$, we have $\sum_{k=0}^{K} \theta_k P_k X = \sum_{k=0}^{K} \sum_{i=0}^{K} \phi_k p_k i M X$, which completes the proof. □

Proof of Proposition 2. Let $\{u_i | i \in [n]\}$ be the orthogonal eigenvectors of $L_U$, where $u_i$ is associated with eigenvalue $\lambda_i$. Then $U g_w (\Lambda) U^\top x = \sum_{i=1}^{n} g_w (\lambda_i) u_i \bar{u}^\top x$. Moreover, since $P$ is the similar matrix of $L_U$, so $P$ owns the same eigenvectors of $L_U$. Let $T$ be a subset of $[n]$ with $|T| = t$ that contains the index of $i \in [n]$ such that $u_i^\top x \neq 0$ for $i \in T$ and $u_i^\top x = 0$ for $i \in n \setminus T$. Then, we have $\sum_{i=0}^{t} \theta_k P_k x = \sum_{i=0}^{t} -\sum_{i \in T} \theta_k u_i^\top x = \sum_{i \in T} ^{t-1} \theta_k u_i^\top x$. Therefore, $\sum_{k=0}^{t-1} \theta_k P_k x = g_w (\lambda_i)$ holds with proper $\theta_k$ for $i \in T$. □

Proof of Theorem 1. The propagation matrix $P$ can be decomposed as $P = U A U^{-1}$ where the $i$-th column of $U$ is the eigenvector of $\lambda_i (P)$, denoted by $\phi_i$. Then we have

$$\lim_{K \to \infty} P^K = \lim_{K \to \infty} (\Phi A \Phi^{-1})^K$$

$$= \lim_{K \to \infty} \Phi A \Phi^{-1}$$

$$= \Phi \lim_{K \to \infty} A \Phi^{-1}$$

$$= \Phi \lim_{K \to \infty} \lambda_K \Phi^{-1}$$

$$= \Phi \lim_{K \to \infty} \lambda_k \Phi^{-1} \cdot \cdots \Phi^{-1}$$

Thus we have $\lim_{K \to \infty} P^K = P_U = \phi_n \cdots \phi_1$ where $\phi_n = \frac{D_1}{\sqrt{d_m}}$. Therefore, we have $P_U [u, v] = \frac{\sqrt{d_u}}{\sqrt{d_v}}$. Let $e_u \in \mathbb{R}^{1 \times n}$ be an indicator vector having 1 in coordinate $u \in V$. Then $P^K [u, v]$ is expressed as $P^K [u, v] = e_u P^K e_v^\top$. For $e_u$ and $e_v$, we decompose $e_u = \sum_{i=0}^{n} \alpha_i \phi_i^\top$, $e_v = \sum_{i=0}^{n} \beta_i \phi_i^\top$. Notice that $\alpha_n = \frac{\sqrt{d_v}}{\sqrt{d_u}}$ and $\beta_n = \frac{\sqrt{d_u}}{\sqrt{d_v}}$. We have

$$\max_{u,v \in V} \frac{|P^K [u, v] - P_U [u, v]|}{P_U [u, v]}$$

$$= \max_{u,v \in V} \frac{|e_u P^K e_v^\top - P_U [u, v]|}{P_U [u, v]}$$

$$\leq \max_{u,v \in V} \frac{\sum_{i=1}^{n-1} |\lambda_i (P) K \alpha_i \beta_i|}{P_U [u, v]}$$

$$\leq \lambda^K \max_{u,v \in V} \frac{\sum_{i=1}^{n-1} |\alpha_i \beta_i|}{P_U [u, v]}$$

$$\leq \lambda^K \max_{u,v \in V} \frac{\|e_u\|\|e_v\| \cdot 2m}{\sqrt{d_u d_v}}$$

$$\leq \lambda^K \frac{2m}{d_{\min}}$$

where $\lambda = \max (-\lambda_1 (P), \lambda_n - 1 (P))$ and $d_{\min} = \min \{d_0 : v \in V\}$. Therefore, if $K \geq \ln \frac{2m}{d_{\min}} / \ln \lambda, \lambda^K \cdot \frac{2m}{d_{\min}} \leq e$, which completes the proof. □

Proof of Theorem 2. W.l.o.g, we set $t = \frac{1-t}{2}$ for $r > 0$. In particular, $t \in (0, 1)$ for $r = 0$ and $t \in [0, 1]$ for $r \in (0, 1)$. Thus, we have $P^r = (D_{+} + 1)^r \frac{1}{2} (A_{+} + 1)(D_{+} + 1)^{-\frac{1}{2}}$. Assume that $v_1, v_2, \ldots, v_{t-1}$ are eigenvectors of $L_U$ corresponding to eigenvalues $\lambda_1 (\tau), \ldots, \lambda_{t-1} (\tau)$, respectively. According to Courant–Fischer theorem [13], we have

$$\lambda_i (\tau) = \min_{x: v_j^\top x = 0 for j \in [1:t-1]} \frac{x^\top (D_{+} + 1)^r \frac{1}{2} (A_{+} + 1)(D_{+} + 1)^{-\frac{1}{2}} x}{x^\top (D_{+} + 1)x}$$

Notice that when $\tau$ increases, $t$ decreases. Consequently, $\lambda_i (\tau)$ increases. Meanwhile, we also get $\lambda_i (\tau) \leq \lambda_i (\tau) = \lambda_i (\tau) \geq \lambda_i (\tau) > \lambda_i (\tau)$ for $r > 1$, which completes the proof. □

Proof of Theorem 3. Let $y \in \mathbb{R}^n$ be the label vector, i.e., $y_u = i$ when node $u$ belongs to the $i$-th class. Let $L_U = U A U^{-1}$ where $U$ is the eigenvector matrix and $A$ is the diagonal matrix of eigenvector spectrum $(\lambda_1, \lambda_2, \ldots, \lambda_n)$. Suppose $U^\top y = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ where $\alpha_i$ is the projection (response) of $y$ on eigenvector $U^\top [i, .]$. Then we have $y^\top L_U y = y^\top U A U^{-1} y = \sum_{i=1}^{n} \alpha_i^2 \lambda_i$.

W.l.o.g, we have $F_1 (x) = \sum_{k=0}^{t-1} w_k P_k x = \sum_{k=0}^{t-1} \sum_{i=1}^{n} \lambda_i w_k \phi_i \phi_i^\top x$ and $F_2 (x) = \sum_{k=0}^{t-1} w_k P_k^2 x = \sum_{k=0}^{t-1} \sum_{i=1}^{n} \lambda_i w_k \phi_i \phi_i^\top x$ where $\lambda_i$ (resp. $\lambda_i (\tau)$) is the $i$-th eigenvalue and $\psi_i$ (resp. $\phi_i$) is the associated eigenvector of $P$ (resp. $P_\tau$). For ease of exposition, we assume $\phi_i^\top x = y_i$ and $\phi_i^\top x = \beta_i$ for $i \in \{1, 2, \ldots, n\}$. Therefore, for the corresponding frequencies of $F_1 (x)$ and $F_2 (x)$, we have

$$\sum_{i=1}^{n} \lambda_i w_k \phi_i \phi_i^\top x = \sum_{i=1}^{n} \lambda_i w_k \beta_i \beta_i^\top x = \sum_{i=1}^{n} \alpha_i^2 \lambda_i \tau$$

(10)

For better demonstration, we represent Equation (10) in the form of

$$(y^\top \phi_1^\top, \ldots, y^\top \phi_n^\top)^\top (w_0^\top, w_1^\top, \ldots, w_{t-1}^\top)$$

$$= (\beta_1, \beta_2, \ldots, \beta_n)^\top \Gamma (w_0^\top, w_1^\top, \ldots, w_{t-1}^\top)$$

$$= (\alpha_1^2 \lambda_1, \alpha_2^2 \lambda_2, \ldots, \alpha_n^2 \lambda_n)^\top \Gamma (\lambda_1, \lambda_2, \ldots, \lambda_n)$$

(11)

where $\Gamma$ (resp. $\Gamma_\tau$) $\in \mathbb{R}^{k \times k}$ is the Vandermonde matrix of $\lambda_i (\tau)$ (resp. $\lambda_i$). Specifically, $\Gamma_\tau = \cdots \lambda_{n-1} (\tau) \lambda_n^{n-2} (\tau) \cdots \lambda_2 (\tau) \lambda_1 (\tau)$.

Note that $\lambda_1 (\tau), \lambda_2 (\tau), \ldots, \lambda_n (\tau)$ and $\Gamma$ in $F_1 (x)$ are constant while $\beta_1, \beta_2, \ldots, \beta_n$ and $\Gamma_\tau$ in $F_2 (x)$ are functions of $\tau$. Meanwhile, as proved in Theorem 2, $\lambda_i (\tau)$ is a monotonically increasing
function of \( \tau \) with controllable values, i.e., either \( \lambda_1(\tau) \leq \lambda_i \) or \( \lambda_i(\tau) > \lambda_i \). In particular, \( F_2(x) \) is a general extension of \( F_1(x) \), e.g., \((\beta_1, \beta_2, \cdots, \beta_n) = (y_1^2, y_2^2, \cdots, y_n^2) \) and \( \Gamma_{+} = \Gamma \) when \( \tau = 1 \).

As a consequence, for a randomly initialized weight \( \mathbf{w} \), let \( \{ \mathbf{w}, \mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_p, \mathbf{w}^i \} \) be the learning trace of a model from \( \mathcal{K}_P(\mathbf{x}) \) to optimal filter \( F_1 \) after training in \((p+1)\) steps. In thus a case, there exists an appropriate \( \tau > 0 \) to alter \( y_i \) into \( \beta_i \) and reshape \( \Gamma \) into \( \Gamma_{+} \) to reach \( \mathbf{w}^i \) before \((p+1)\) steps, ensuring Equation (11). Put differently, obtaining \( \mathbf{w}^i \) is more attainable than reaching \( \mathbf{w}^o \) starting from the initial weight \( \mathbf{w} \) in expectation, i.e.,

\[
\mathbb{E} [\| \mathbf{w}^o - \mathbf{w}^i \|_2] \leq \mathbb{E} [\| \mathbf{w}^o - \mathbf{w} \|_2].
\]

Table 6: Hyperparameters of AdaptKry.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Hop K</th>
<th>( \tau )</th>
<th>Learning rate</th>
<th>Hidden dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>10</td>
<td>(0.5, 0.8, 1.1)</td>
<td>0.10</td>
<td>256</td>
</tr>
<tr>
<td>Citeeseer</td>
<td>10</td>
<td>0.1</td>
<td>0.01</td>
<td>128</td>
</tr>
<tr>
<td>Pubmed</td>
<td>10</td>
<td>0.5</td>
<td>0.10</td>
<td>256</td>
</tr>
<tr>
<td>Actor</td>
<td>10</td>
<td>(0.6, 1.7, 1.8)</td>
<td>0.10</td>
<td>256</td>
</tr>
<tr>
<td>Chameleon</td>
<td>10</td>
<td>(0.5, 0.8)</td>
<td>0.01</td>
<td>256</td>
</tr>
<tr>
<td>Squirrel</td>
<td>10</td>
<td>0.8</td>
<td>0.005</td>
<td>256</td>
</tr>
</tbody>
</table>

A.2 Experimental Settings

Running Environment. All experiments are conducted on a Linux machine with an NVIDIA Tesla V100 (32GB memory), Intel Xeon(R) CPU (2.80GHz), and 500GB RAM.

Parameter Settings. Table 6 presents the hyperparameters (hop \( K \), \( \tau \), learning rate and hidden dimension) of AdaptKry adopted on the 6 small datasets and 4 large datasets. As said in Section 5.1, we fix \( K = 10 \) for the small datasets and tune \( K \) in \{2, 4, 6, 8, 10\} for the large datasets by following the setting in [20]. For \( \tau \), we tune \( \tau \) in the range [0.1, 1]. In particular, we normally set \( \tau = 0.9 \) for the three strong heterophily graphs, as verified in our experiment (Section 5.3), and tune it for the rest datasets for the best fit. Learning rate (lr) is selected from the set \{0.005, 0.01, 0.05, 0.1\} and hidden dimension is tuned from the set \{128, 256, 512, 1024, 2048\}. For baselines, we either adopt their recommended parameter settings on corresponding datasets or tune their parameters carefully following the above scheme for a fair comparison.

Implementation Details. We implement AdaptKry in PyTorch. The baselines are obtained from their official release. Table 7 summarizes the links to baseline codes. As with ChebNet and GPR-GNN, we adopt their implementations inside the source code of BernNet.

Table 7: Download links of baseline methods.

<table>
<thead>
<tr>
<th>Methods</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td><a href="https://github.com/pyg-team/pytorch_geometric">https://github.com/pyg-team/pytorch_geometric</a></td>
</tr>
<tr>
<td>SGC</td>
<td><a href="https://github.com/Tiiiger/SGC">https://github.com/Tiiiger/SGC</a></td>
</tr>
<tr>
<td>ASGC</td>
<td><a href="https://github.com/schariya/adaptive-simple-convolution">https://github.com/schariya/adaptive-simple-convolution</a></td>
</tr>
<tr>
<td>SIGN</td>
<td><a href="https://github.com/twitter-research/sign">https://github.com/twitter-research/sign</a></td>
</tr>
<tr>
<td>BernNet</td>
<td><a href="https://github.com/ivam-he/BernNet">https://github.com/ivam-he/BernNet</a></td>
</tr>
<tr>
<td>JacobConv</td>
<td><a href="https://github.com/GraphPKU/JacobConv">https://github.com/GraphPKU/JacobConv</a></td>
</tr>
<tr>
<td>EvenNet</td>
<td><a href="https://github.com/leirunlin/evenet">https://github.com/leirunlin/evenet</a></td>
</tr>
<tr>
<td>ChebNetII</td>
<td><a href="https://github.com/ivam-he/ChebNetII">https://github.com/ivam-he/ChebNetII</a></td>
</tr>
<tr>
<td>OptBasisGNN</td>
<td><a href="https://github.com/yuziGuo/FarOptBasis">https://github.com/yuziGuo/FarOptBasis</a></td>
</tr>
</tbody>
</table>

Figure 5: Non-orthogonal and orthogonal polynomial computation overheads.

Further exploration on orthogonality. For a complete evaluation, we further test AdaptKry and OrthKry on datasets Pubmed and Actor to compare non-orthogonal and orthogonal polynomials in graph filters. Table 8 presents the accuracy scores on the homophily dataset Pubmed and the heterophily dataset Actor. We observe that AdaptKry shows a clear advantage over OrthKry on Pubmed. Except for the case of \( K = 2 \), AdaptKry also beats OrthKry on heterophily dataset Actor. Together with the results in Table 5, neither AdaptKry nor OrthKry graph filter demonstrates dominance over the other across various graphs, and our experiments indicate that orthogonal polynomials do not exhibit clear advantages over non-orthogonal polynomials.

Meanwhile, we also plot the polynomial computation overheads of AdaptKry and OrthKry on datasets Pubmed and Actor in Figure 5. Similar to the results in Figure 4, the computation overheads of non-orthogonal polynomials in AdaptKry can be one order of magnitude smaller than those of orthogonal polynomials in OrthKry. Those findings substantiate the superior efficacy of non-orthogonal bases in comparison to orthogonal bases.

Learned graph filters. To better illustrate the graph filtering capability of AdaptKry, we plot the graph filters learned on the 3 homophily graphs (Cora, Citeeseer, Pubmed) and 3 heterophily graphs (Actor, Chameleon, Squirrel) in Figure 6. As seen, AdaptKry learns explicit low-pass filters on homophily graphs and a high-pass filter on Actor, comb-pass filters on Chameleon and Squirrel, coherent with the results in the literature [21].
Figure 6: Graph filters learnt by AdaptKry on 6 graphs with various homophily ratios