Accelerated filtering on graphs using Lanczos method

Ana Šušnjara[1], Nathanaël Perraudin[2], Daniel Kressner[1], and Pierre Vandergheynst[2]

[1] ANCHP
Ecole Polytechnique Fédérale de Lausanne EPFL

[2]LTS2
Ecole Polytechnique Fédérale de Lausanne EPFL

Abstract

Signal-processing on graphs has developed into a very active field of research during the last decade. In particular, the number of applications using frames constructed from graphs, like wavelets on graphs, has substantially increased. To attain scalability for large graphs, fast graph-signal filtering techniques are needed. In this contribution, we propose an accelerated algorithm based on the Lanczos method that adapts to the Laplacian spectrum without explicitly computing it. The result is an accurate, robust, scalable and efficient algorithm. Compared to existing methods based on Chebyshev polynomials, our solution achieves higher accuracy without increasing the overall complexity significantly. Furthermore, it is particularly well suited for graphs with large spectral gaps. [1]

Index terms— Spectral graph theory, graph signal-processing, graph filter, Chebyshev polynomial, Lanczos method

1 Introduction

Graphs conveniently represent data on irregular geometric structures as they arise in numerous application domains such as social, energy, transportation or neuronal networks. In all of these fields, different pieces of information are connected to each other and these connections can be modelled by a graph. For every link, an edge is drawn with an associated weight that represents the similarity between the two elements (vertices) it connects. For example, in a sensor network acquiring environmental measurements, it makes sense to choose the weight inversely proportional to the physical distance. A data signal lives on nodes and can be visualized as a collection of samples. We refer to those as graph-signals, see Figure 1 for an example.

This framework, called graph signal-processing, has recently become a general field of research [16, 17] and applications of graph signal-processing can be found in many different areas. In machine vision, automatic text classification or more generally

[1]The work of A. Šušnjara has been supported by the SNSF research project *Low-rank updates of matrix functions and fast eigenvalue solvers*. The work of Nathanaël Perraudin has been supported by the SNF research project *Towards Signal Processing on Graphs*, grant number: 2000_21/154350/1 .

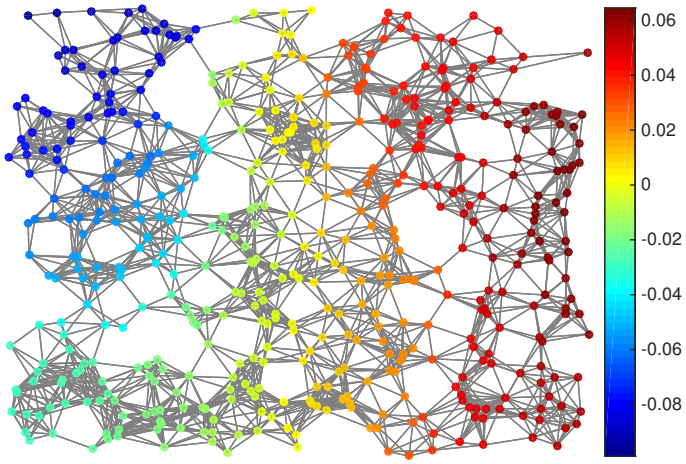

Figure 1: Synthesized sensor network. Colored circles represent vertices and gray links represent edges. Values of the signal are displayed with different colors.

machine learning, graphs are used to represent similarities between data points and result in algorithms for semi-supervised learning [24, 19, 1, 23]. Graph signal-processing is often used to leverage intrinsic links when processing data. For instance, many applications in image processing benefit from the use of graphs that describe connections between local patches in the image (e.g. [13, 22] and references therein).

Spectral graph filtering is one of the basic building blocks in the applications discussed above. Classical filtering techniques rely on the Fourier transform, which can be done inexpensively with a cost of $\mathcal{O}(N \log(N))$. Extending this transform to graphs requires the diagonalisation of the graph Laplacian, which becomes prohibitively expensive for larger graphs. To overcome this issue, an efficient approximate method to perform spectral graph filtering was proposed in [10]. This method, based on Chebyshev polynomial approximation, only requires multiple application of the Laplacian operator and thus leads to a scalable, efficient memory-saving and error-controlled algorithm.

Instead of Chebyshev polynomial approximation, we propose to use the Lanczos method to perform spectral graph filtering. The resulting algorithm is also scalable, in addition to being superior in accuracy to its predecessor.

The rest of this article is organized as follows. In Section 2, we summarize the basics of signal-processing on graphs, including the definition of graph filtering. We also recall the method of Hammond et al. [10]. Section 3 describes the Lanczos method and its application to graph filtering. Numerical experiments are presented in Section 4.

2 Signal-processing on graphs

2.1 Graph nomenclature

We consider a weighted undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ with a set of vertices $\mathcal{V}$, a set of edges $\mathcal{E}$, and a weight function $\mathcal{W} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$. The vertices are indexed from $1, \dots, N = |\mathcal{V}|$ and each entry of the weight matrix $W \in \mathbb{R}^{N \times N}$ contains the weight of the edge connecting the corresponding vertices: $W_{i,j} = \mathcal{W}(v_i, v_j)$. If there is no edge between two vertices, the weight is set to 0. It is assumed that $W_{i,j} = W_{j,i}$, that is, W is a symmetric matrix. For a vertex $v_i \in \mathcal{V}$, the degree $d(i)$ is defined as the sum of the weights of incident edges: $d(i) = \sum_{j=1}^{N} W_{i,j}$.

In this framework, a graph signal is defined as a function $s : \mathcal{V} \to \mathbb{R}$ assigning a value to each vertex. It is convenient to consider a signal s as a vector of size N with the i^{th} component representing the signal value at the i^{th} vertex.

One of the most fundamental concepts for weighted undirected graphs is the (combinatorial) graph Laplacian $\mathcal{L}$ defined as $\mathcal{L} = D - W$, where D is the diagonal degree matrix with diagonal entries $D_{ii} = d(i)$. Alternative definitions of graph Laplacians include the normalized Laplacian $\mathcal{L}_n = D^{\frac{1}{2}} \mathcal{L} D^{\frac{1}{2}} = D^{\frac{1}{2}}(D - W)D^{\frac{1}{2}}$. For simplicity, we restrict ourselves to $\mathcal{L}$, but the method presented in this paper easily extends to other Laplacians.

The Laplacian $\mathcal{L}$ is always symmetric positive semi-definite and can thus be decomposed as

$$\mathcal{L} = U \Lambda U^*,$$

where $U = [u_0, \dots, u_{N-1}]$ is an orthogonal matrix and U^* denotes its transpose. Without loss of generality, we order the set of eigenvalues as follows: $0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{N-1} = \lambda_{\max}$; see [2] for more details on spectral graph theory. The matrix U defines the graph Fourier basis [16, 17], leading to the graph Fourier transform $\hat{s} = U^* s$ and its inverse $s = U\hat{s}$.

2.2 Graph filters

In the classic setting, applying a filter to a signal is performed by convolution, which corresponds to point-wise multiplication in the spectral domain. Similarly, filtering a graph-signal is peformed by multiplication with a filter in the graph Fourier domain. A graph filter is defined by a continuous function $g : \mathbb{R}_+ \to \mathbb{R}$. To obtain its discrete coefficients, this function is evaluated at each eigenvalue: $g(\lambda_\ell)$ for $\ell = 0, \dots, N-1$. The filtering operation then corresponds to $\hat{s}'(\ell) = g(\lambda_\ell) \cdot \hat{s}(\ell)$, where s' is the filtered signal. Equivalently, using matrix notation, we have

$$s' = U g(\Lambda) U^* s, \tag{1}$$

where $g(\Lambda)$ is the diagonal matrix containing the coefficients $g(\lambda_\ell)$ on the diagonal. In terms of matrix functions [11], the relation (1) can be compactly expressed as $s' = g(\mathcal{L})s$ with $g(\mathcal{L}) := U g(\Lambda) U^*$.

2.3 Fast filtering via Chebyshev polynomials

The graph filtering operation described above is based on the graph Fourier transform. Unfortunately, the graph Fourier basis needed for performing this transform requires the diagonalization of the graph Laplacian, which takes $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ memory when using standard techniques. This is feasible for graphs with only a few thousand vertices. To be able to tackle problems of larger size, more efficient methods are needed, one of which is presented in [10] and summarized in this section.

Filtering in the vertex domain. To avoid the Fourier transform, we perform the filtering operation in the vertex domain using only the Laplacian operator. Applying this operator corresponds to multiplying the signal in the spectral domain with the eigenvalues:

$$\widehat{\mathcal{L}s} = \Lambda \hat{s}.$$

This is equivalent to filtering with $g(x) = x$. Using this relation recursively and exploiting linearity, we can apply any polynomial filter $g(x) = a_0 + a_1 x + \cdots + a_K x^M$ to a signal s with the following formula:

$$s' = U g(\Lambda) U^* s = \left(a_0 I + a_1 \mathcal{L} + \cdots + a_M \mathcal{L}^M\right) s. \tag{2}$$

Chebyshev polynomial approximation. The ability to apply polynomial filters efficiently suggests to approximate a given filter function with a suitable polynomial. For approximating functions on real intervals, Chebyshev polynomials are usually the preferred choice because of numerical stability considerations and the fact that they can be evaluated efficiently by three-term recurrences. We refer to [10] for a more detailed discussion on the choice of Chebyshev polynomials in signal-processing on graphs and to, e.g., [14] for an introduction to polynomial approximation.

The mth Chebyshev polynomial $T_m(y)$ is generated using the recurrence relation $T_m(y) = 2xT_{m-1}(y) - T_{m-1}(y)$ with $T_0(y) = 1$ and $T_1(y) = y$. For $y \in [-1, 1]$, these polynomials possess the following well-known properties:

1. they admit the closed form expression $T_m(y) = \cos(m \arccos(y))$;
2. they are bounded, i.e., $T_m(y) \in [-1, 1]$;
3. they form an orthogonal basis of $L^2\left([-1,1], \frac{\mathrm{d}y}{\sqrt{1-y^2}}\right)$.

The third property implies that every function $h \in L^2\left([-1,1], \frac{\mathrm{d}y}{\sqrt{1-y^2}}\right)$ admits a convergent Chebyshev series

$$h(y) = \frac{1}{2}c_0 + \sum_{m=1}^{\infty} c_m T_m(y),$$

with the Chebyshev coefficients

$$c_k = \frac{2}{\pi}\int_{-1}^{1} \frac{T_m(y)h(y)}{\sqrt{1-y^2}}\mathrm{d}y = \frac{2}{\pi}\int_0^{\pi} \cos(k\theta)h(\cos(\theta))\mathrm{d}\theta.$$

Since our filter g is evaluated on the eigenvalues of the Laplacian, we need to map the interval $[-1, 1]$ to the interval $[0, \lambda_{\max}]$ using the transformation $x = \frac{\lambda_{\max}}{2}(y + 1)$. Defining $\tilde{T}_m(x) = T_m\left(\frac{2x}{\lambda_{\max}} - 1\right)$ we obtain

$$g(x) = \frac{1}{2}c_0 + \sum_{m=1}^{\infty} c_m \tilde{T}_m(x) \tag{3}$$

for $x \in [0, \lambda_{\max}]$, with

$$c_m = \frac{2}{\pi}\int_0^{\pi} \cos(m\theta) g\left(\frac{\lambda_{\max}}{2}(\cos(\theta) + 1)\right) \mathrm{d}\theta.$$

Fast filtering algorithm. At this point, we can derive the iterative algorithm for filtering a signal s with g. The recurrence relation for the transformed Chebyshev polynomials becomes $\tilde{T}_m(x) = 2\left(\frac{2x}{\lambda_{\max}} - 1\right)\tilde{T}_{m-1}(x) - \tilde{T}_{m-2}(x)$. On the matrix level, this yields, using (2):

$$\tilde{T}_m(\mathcal{L})s = 2\left(\frac{2\mathcal{L}}{\lambda_{\max}} - I\right)\tilde{T}_{m-1}(\mathcal{L})s - \tilde{T}_{m-2}(\mathcal{L})s.$$

Combined with (3), this finally leads to the following expression for filtering a signal s:

$$s' = g(\mathcal{L}) = \frac{1}{2}c_0 I s + \sum_{m=1}^{\infty} c_m \tilde{T}_m(\mathcal{L})s.$$

When implemented, we truncate this sum at a defined order M. Assuming that $|\mathcal{E}| > N$, the computational cost of this algorithm scales linearly with the number of edges $\mathcal{O}(M|\mathcal{E}|)$. In most applications, the Laplacian is sparse, $|\mathcal{E}| \ll N^2$, which results in a fast algorithm. Moreover, apart from storing the Laplacian, the additional memory consumed by this algorithm is only $4N$.

3 Accelerated filtering using Lanczos

Given the graph Laplacian $\mathcal{L} \in \mathbb{R}^{N\times N}$ and a nonzero vector $s \in \mathbb{R}^N$, the Lanczos method [7] shown in Algorithm 1 below computes an orthonormal basis $V_M = [v_1, \ldots, v_M]$ of the Krylov subspace $K_M(\mathcal{L}, s) = \mathrm{span}\{s, \mathcal{L}s, \ldots, \mathcal{L}^{M-1}s\}$. The computational cost of Algorithm is $\mathcal{O}(M \cdot |\mathcal{E}|)$. The storage of the basis V_M requires MN additional memory, which can be avoided using two passes of the algorithm or restart techniques; see, e.g., [5] for more details.

The scalars produced by Algorithm 1 can be arranged into a symmetric tridiagonal matrix $H_M \in \mathbb{R}^{M\times M}$ satisfying

$$V_M^* \mathcal{L} V_M = H_M = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_M \\ & & & \beta_M & \alpha_M \end{bmatrix}.$$

Algorithm 1 Lanczos method

Input: Symmetric matrix $\mathcal{L} \in \mathbb{R}^{N \times N}$, vector $s \neq 0$, $M \in \mathbb{N}$.

Output: $V_M = [v_1, \ldots, v_M]$ with orthonormal columns, scalars $\alpha_1, \ldots, \alpha_M \in \mathbb{R}$ and $\beta_2, \ldots, \beta_M \in \mathbb{R}$.

$v_1 \leftarrow s/\|s\|_2$
for $j = 1, 2, \ldots, M$ **do**
  $w = \mathcal{L}v_j$
  $\alpha_j = v_j^* w$
  $\tilde{v}_{j+1} = w - v_j \alpha_j$
  if $j > 1$ **then**
    $\tilde{v}_{j+1} \leftarrow \tilde{v}_{j+1} - v_{j-1}\beta_{j-1}$
  end if
  $\beta_j = \|\tilde{v}_{j+1}\|_2$
  $v_{j+1} = \tilde{v}_{j+1}/\beta_j$
end for

In floating-point arithmetics, the orthogonality of the basis produced by Algorithm 1 may get quickly lost and reorthogonalization is needed [3].

Given a continuous function $g : [0, \lambda_{\max}] \to \mathbb{R}$ and a vector s, the following approximation to $g(\mathcal{L})s$ was proposed by Gallopoulos and Saad in [6]:

$$g(\mathcal{L})s \approx \|s\|_2 V_M g(H_M) e_1 \coloneqq g_M, \tag{4}$$

where $e_1 \in \mathbb{R}^M$ is the first unit vector. Because of eigenvalue interlacing, the eigenvalues of H_M are contained in the interval $[0, \lambda_{\max}]$ and hence the expression $g(H_M)$ is well-defined. Typically, $M \ll N$, rendering the evaluation of $g(H_M)$ inexpensive. The overall cost of our Lanczos-based approximation of graph-signal filtering, which consists of applying Algorithm 1 and evaluating (4), is therefore between $O(M \cdot |\mathcal{E}|)$ and $O(M \cdot |\mathcal{E}| + M^2 N)$, depending on how the reorthogonalization is performed.

Approximation quality. The following theorem provides some insight into the accuracy of the approximation g_M obtained by the Lanczos method.

Theorem 1 ([9, Corollary 3.4])**.** *Let $\mathcal{L} \in \mathbb{R}^{N \times N}$ be symmetric with eigenvalues contained in the interval $[0, \lambda_{\max}]$ and let $g : [0, \lambda_{\max}] \to \mathbb{R}$ be continuous. Then*

$$\|g(\mathcal{L})s - g_M\|_2 \leq 2\|s\|_2 \cdot \min_{p \in \mathcal{P}_{M-1}} \max_{z \in [0, \lambda_{\max}]} |g(z) - p(z)|,$$

where $\mathcal{P}_{M-1}$ denotes all polynomials of degree at most $M - 1$.

Theorem 1 shows that the error is bounded by the best polynomial approximation [14] of g on $[0, \lambda_{\max}]$. Compared to the Chebyshev approximation from Section 2.3, this shows that – up to a multiple of two – the Lanczos-based approximation g_M can be expected to provide at least the same accuracy.

However, the Lanczos-based approximation can sometimes be expected to perform significantly better because of its ability to adapt to the eigenvalues of $\mathcal{L}$. This phenomenon is well-understood for Krylov subspace approximations to solutions of linear

systems (see, e.g., [8, Section 3.1]) and extends to matrix functions as well. To illustrate such a situation, suppose that the eigenvalue 0 of $\mathcal{L}$ is well separated from the other eigenvalues contained in the (smaller) interval $[\lambda_1, \lambda_{\max}]$. Then the eigenvalues of H_M can be expected to exhibit a similar behavior [15, Chapter 6].

By choosing a polynomial that interpolates the eigenvalue 0 exactly and approximates g on $[\lambda_1, \lambda_{\max}]$, the bound of Theorem 1 can approximately be replaced by

$$\|g(\mathcal{L})s - g_M\|_2 \lesssim 2\|s\|_2 \cdot \min_{p \in \mathcal{P}_{M-2}} \max_{z \in [\lambda_1, \lambda_{\max}]} |g(z) - p(z)|.$$

At the expense of losing a degree of freedom in the choice of the polynomial, the width of the approximation interval becomes smaller, which in turn leads to significantly improved approximation rates [14]. In contrast, the Chebyshev approximation cannot adapt to the eigenvalues of $\mathcal{L}$ and hence its approximation rate is driven by the larger interval $[0, \lambda_{\max}]$.

Stopping criteria. Ideally, M should be chosen to be the smallest integer such that

$$\|e_M\|_2 = \|g(\mathcal{L})s - g_M\|_2 \leq \epsilon$$

holds for some prescribed accuracy $\epsilon > 0$. As proposed in [20], we estimate e_M as the difference of two (consecutive) approximations:

$$e_M \approx g_{M+j} - g_M \tag{5}$$

for a small value of j.

4 Numerical experiments

In this section we present numerical experiments to demonstrate the numerical behaviour of the Lanczos method and compare it with the Chebyshev method. All experiments were performed with the signal processing toolbox GSPBox [12] and can be downloaded at `https://lts2.epfl.ch/lanczos-filtering/`.

Example 1. We first consider the sensor graph and the Erdős–Rényi random graph [4], with $N = 500$ nodes. In the latter case, each edge is included in the graph with probability $p = 0.04$. As a filterbank, we use a collection of translated windows $g(t) = \sin(0.5\pi \cos(\pi t)^2)\chi_{[-1/2,1/2]}$, where χ_I is the characteristic function of I [21]. We choose to adapt the filterbank to the graph spectrum, since non-adapted filters lead to very coherent frames for graphs with eigenvalues not uniformly spread along the spectrum [18].

First, we test the reliability of the error estimate (5) for $j = 3$. Figure 2 confirms that the estimate $\|g_{M+3} - g_M\|_2$ closely follows the true error $\|e_M\|_2$.

In Figures 3 and 5 we observe that the Lanczos method yields better approximation, especially in the case when the spectrum of graph Laplacian is not uniformly distributed and has a large (relative) spectral gap, as predicted in Section 3.

Example 2. The purpose of this example is to investigate how the approximation by the Chebyshev polynomial method and the Lanczos method depends on the edge sparsity of the Erdős – Rényi graph. We consider graphs with $N = 1000$ nodes.

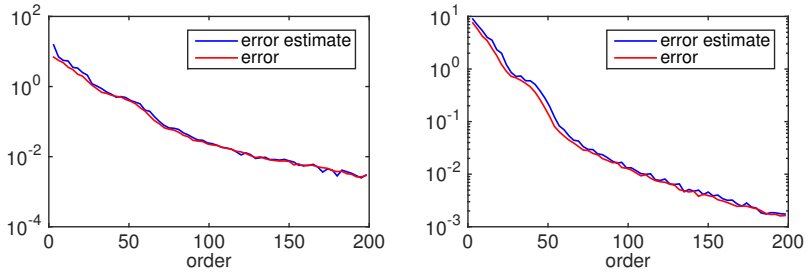

Figure 2: Sensor graph (left) and Erdős – Rényi graph (right). Comparison of the approximation error in the Lanczos method and the estimate (5).

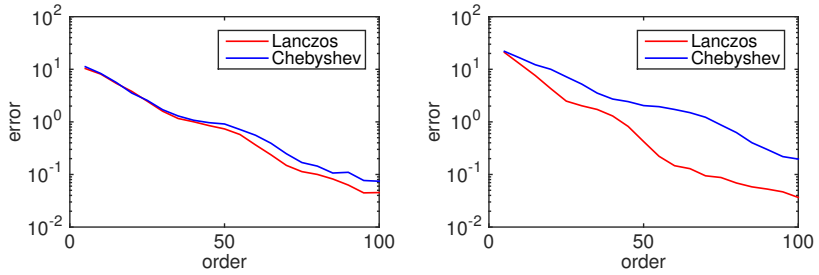

Figure 3: Sensor graph (left) and Erdős – Rényi graph (right). Comparison of the approximation error using the Chebyshev method and the Lanczos method with respect to the order of approximation.

The order of Chebyshev polynomial and Lanczos method are set to $M = 30$. As a filterbank, we the use mexican hat wavelet, with a mother window

$$g_h(\lambda_\ell) = \lambda_\ell \cdot \exp(-\lambda_\ell^2),$$

with λ_ℓ eigenvalues of graph Laplacian. In order to obtain a complete filterbank, we add a low pass filter $g_l(\lambda_\ell) = \exp(-\lambda_\ell^4)$.

We notice that the Lanczos method exhibits excellent numerical behaviour in comparison with the Chebyshev method, particularly when using non-adapted filters. As the probability p increases, the (relative) spectral gap increases as well, leading to more accurate Lanczos approximation (see Figure 6).

5 Conclusion

In this letter, a scalable, efficient and accurate method to perform signal filtering on graphs based on the Lanczos method is proposed. We compare it to the existing method based on the Chebyshev polynomial approximation. The advantage of filtering using the Lanczos approximation is that no a priori information about the spectrum of $\mathcal{L}$ is

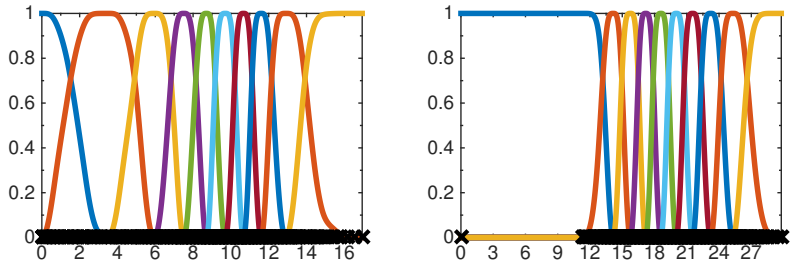

Figure 4: Sensor graph (left) and Erdős – Rényi graph (right). Filterbanks adapted to spectra of graph Laplacians.

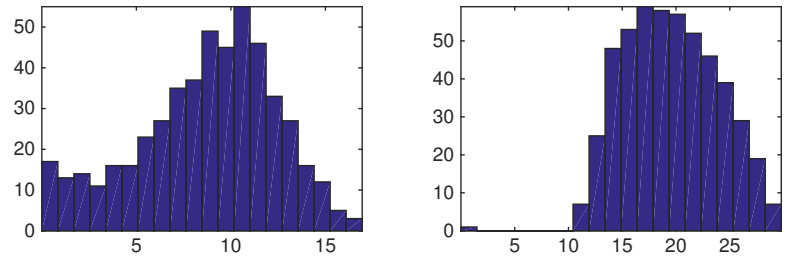

Figure 5: Sensor graph (left) and Erdős – Rényi graph (right). Distribution of graph Laplacians' eigenvalues.

needed. In all numerical experiments the proposed method outperforms the Chebyshev approach. That is especially visible in case of a non-adapted filterbank. Futhermore, the Lanczos approximation is superior to the Chebyshev approximation when filtering on random graphs with well separated eigenvalues.

References

[1] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Learning theory*, pages 624–638. Springer, Berlin Heidelberg, 2004.

[2] F. R. K. Chung. *Spectral graph theory*. Published for the Conference Board of the Mathematical Sciences, Washington, DC; by the American Mathematical Society, Providence, RI, 1997.

[3] J. K. Cullum and R. A. Willoughby. *Lanczos algorithms for large symmetric eigenvalue computations. Vol. 1*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.

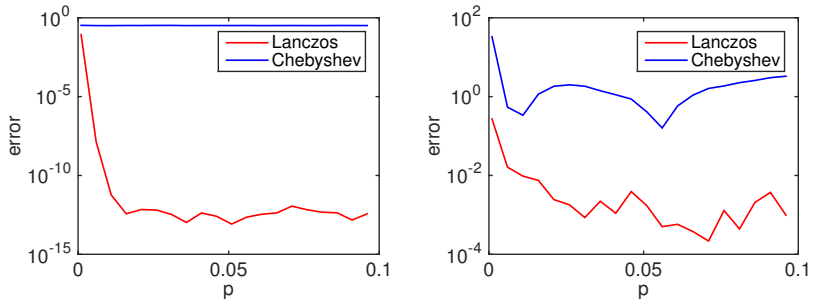

Figure 6: Comparison of the approximation error using the Chebyshev method and the Lanczos method with respect to p, with non-adapted filterbank (left) and adapted filterbank (right).

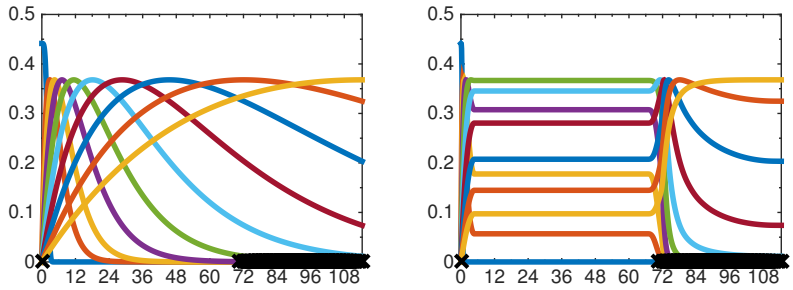

Figure 7: Non-adapted filterbank (left) and adapted filterbank (right).

[4] P. Erdős and A. Rényi. On random graphs. *Publ. Math. Debrecen*, 6:290–297, 1959.

[5] A. Frommer, S. Güttel, and M. Schweitzer. Efficient and stable Arnoldi restarts for matrix functions based on quadrature. *SIAM J. Matrix Anal. Appl.*, 35(2):661–683, 2014.

[6] E. Gallopoulos and Y. Saad. Efficient solution of parabolic equations by Krylov approximation methods. *SIAM J. Sci. Statist. Comput.*, 13(5):1236–1264, 1992.

[7] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, Fourth edition, 2013.

[8] A. Greenbaum. *Iterative methods for solving linear systems*, volume 17 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[9] S. Güttel. Rational Krylov approximation of matrix functions: numerical methods and optimal pole selection. *GAMM-Mitt.*, 36(1):8–31, 2013.

[10] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.*, 30(2):129–150, 2011.

[11] N. J. Higham. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008.

[12] N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. arXiv:1408.5781, 2014.

[13] G. Peyré, S. Bougleux, and L. Cohen. Non-local regularization of inverse problems. In *Computer Vision–ECCV 2008*, pages 57–68. Springer, Berlin Heidelberg, 2008.

[14] G. M. Phillips. *Interpolation and approximation by polynomials*. Springer, New York, 2003.

[15] Y. Saad. *Numerical Methods for Large Eigenvalue Problems: Revised Edition*. Classics in Applied Mathematics 66. SIAM, 2011.

[16] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, 30(3):83–98, 2013.

[17] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. arXiv:1307.5708, 2013.

[18] D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst. Spectrum-adapted tight graph wavelet and vertex-frequency frames. arXiv:1311.0897, 2013.

[19] A. J. Smola and R. Kondor. Kernels and Regularization on Graphs. In *Learning Theory and Kernel Machines*, pages 144–158. Springer, Berlin Heidelberg, 2003.

[20] J. van den Eshof and M. Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM J. Sci. Comput.*, 27(4):1438–1457, 2006.

[21] E. Wesfreid and M. V. Wickerhauser. Adapted local trigonometric transforms and speech processing. *IEEE Trans. Signal Process.*, 41(12):3596–3600, 1993.

[22] F. Zhang and E. R. Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recogn.*, 41(11):3328–3342, 2008.

[23] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 321–328, 2003.

[24] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.