International Journal of Development Mathematics Vol 2 Issue 3 Page No. 200 - 210

INTERNATIONAL JOURNAL OF DEVELOPMENT MATHEMATICS

ISSN: 3026-8656 (Print) | 3026-8699 (Online)

journal homepage: https://ijdm.org.ng/index.php/Journals

# An Adaptive Spectral Method for Dynamic Graphs in Enhancing Numerical Stability and Efficiency

**Olayiwola Babarinsa[a*], Christie Ishola[b], Folaranmi Rotimi[c] and Rasheedat Ayinla-Rahmon[d]**

[a]Department of Mathematical Sciences, Federal University Lokoja, P.M.B 1154, Kogi, Nigeria.

[b,d]Department of Mathematics, National Open University of Nigeria, Jabi-Abuja, Nigeria

[c]Department of Mathematical and Computing Science, Thomas Adewumi University, Kwara State, Nigeria

**ARTICLE INFO**

***Article history:***
Received 11 June 2025
Received in revised form 10 August 2025
Accepted 20 August 2025

**Keywords:**
Spectral Graph Theory, Numerical Stability, Dynamic Systems, Eigensolvers, Laplacian Matrix

**MSC 2020 Subject classification:**
05C50, 65F15, 65Y20, 68R10

**ABSTRACT**

Graph theory and numerical analysis meet in numerous real-world applications, including network evolution, dynamic data clustering, and multiscale modelling. Current spectral methods, however, often presume static graph structures, compromising their accuracy and efficiency for developing networks. This work presents an adaptive spectral technique that adjusts solver tolerance dynamically according to structural changes in the graph, quantified through the Frobenius norm of differences between consecutive Laplacians. We tested this on synthetic graphs and compared it with conventional fixed-tolerance techniques. The adaptive method exhibited runtime improvements of up to 50% and substantially reduced eigenvalue errors. These findings attest to its superior numerical stability and computational efficiency. We advocate the application of adaptive tolerance methods in real-time spectral analysis applications and propose follow-up work develop this framework for weighted, directed, or real-world dynamic networks.

## 1. Introduction

Graph theory has long been a fundamental pillar of discrete mathematics, with applications ranging from theoretical computer science to applied engineering disciplines (Babarinsa, 2022; Bondy & Murty, 2008). When merged with numerical analysis, a discipline focused on the development and stability of algorithms for continuous mathematical problems, graph-based methods provide a powerful computational framework for modelling and solving complex problems in high-dimensional and dynamic domains, which may involve some matrix factorizations (Saad, 2011) .

Matrix factorization has a significant connection between numerical analysis and spectral graph theory. The structure of networks in graph-based settings is encoded by adjacency and Laplacian matrices, and key graph features like connectivity, clustering tendencies, and signal smoothness

* Corresponding author. Tel.: +2348060032554
E-mail address: olayiwola.babarinsa@fulokoja.edu.ng (Babarinsa Olayiwola)
https://doi.org/10.62054/ijdm/0203.14

can be extracted using their factorizations, including eigen-decompositions, singular value decompositions (SVD), WH factorization, and low-rank approximations, see (O Babarinsa & H Kamarulhaili, 2017; Babarinsa & Kamarulhaili, 2018; Chi, Song, Zhou, Hino, & Tseng, 2007; Psorakis, Roberts, Ebden, & Sheldon, 2011; Von Luxburg, 2007) and the references therein. Additionally, matrix factorization approaches facilitate the fast updating of low-rank approximations in time-evolving networks by serving as a link between dynamic system modelling and static graph representations. This demonstrates the growing importance of integrating graph theory and numerical linear algebra to solve problems in real-time analytics, data-driven modelling, and scalable computation.

The field of spectral graph theory, which studies the properties of the Laplacian and adjacency matrices of graphs through their eigenvalues and eigenvectors, is useful in data clustering for semi-supervised learning, community detection, dimensionality reduction via Laplacian eigenmaps and graph partitioning in parallel computing, see (Olayiwola Babarinsa & Hailiza Kamarulhaili, 2017; Belkin & Niyogi, 2003; Chung, 1997; Pothen, Simon, & Liou, 1990; Spielman, 2010; Zhou & Schölkopf, 2004). These spectral techniques rely heavily on the accurate and efficient computation of eigenpairs of large, sparse matrices, tasks typically handled by iterative numerical solvers like the Lanczos algorithm (for symmetric matrices) and the Arnoldi method (for general matrices) have become standard in approximating extremal eigenvalues efficiently (Saad, 2011; Trefethen & Bau, 2022). Most traditional spectral techniques assume static graph structures, which is a limitation in the case of dynamic or evolving graphs (Chung, 1997; Von Luxburg, 2007).

The effect of graph dynamics on spectral quality has become practically relevant for applications in social networks, sensor streams, and time-series interaction graphs. Dong, Thanou, Frossard, and Vandergheynst (2016) demonstrated, with the help of a graph Laplacian learning algorithm, that even small structural changes can substantially perturb the spectrum of the Laplacian. This can cause eigenvalue drifts, rendering fixed-tolerance iterative solvers computationally inefficient or numerically unstable. Their research paved the way for structure-aware eigensolvers, where spectral updates are directly coupled with topological evolution.

In addition, Ipsen and Nadler (2009) suggested a method for computing eigenvalues that

combines matrix perturbation theory and residual norms. Their work highlighted that eigensolvers should adjust based on the predicted spectrum shift caused by matrix perturbations and the residual error of computed eigenpairs, rather than imposing a uniform convergence tolerance. In situations where spectrum changes are minimal, this approach prevents over-computation while guaranteeing improved numerical stability.

However, a crucial issue arises when these methods are applied to dynamic graphs whose topology evolves. In practice, such as social networks, financial systems, and sensor networks, the underlying graph can change continuously due to additions or deletions of nodes and edges. These changes, even if small, can significantly alter the spectral structure of the graph, and thus, numerical solutions obtained without accounting for such evolution may become inaccurate or computationally expensive to maintain (Dong *et al.*, 2016).

Existing methods for spectral analysis often assume a static graph structure or use fixed convergence tolerances in iterative solvers (Meyer, 2023). Such assumptions are not only restrictive but can also lead to inefficiencies: either by over-computing for negligible changes or by under-solving for significant transitions in graph structure. Error-controlled eigensolvers that incorporate matrix perturbation theory and residual estimation into their convergence criteria offer a promising pathway (Ipsen & Nadler, 2009). This motivates the need for a dynamic framework that adjusts its computational rigour following the temporal changes in the graph (Pasham, 2024).

In this paper, we propose an adaptive spectral method designed for numerically stable and efficient computation on synthetic dynamic graphs. This enables an error tolerance scheme that adapts to the rate of structural change in the graph, measured using the Frobenius norm of the difference between successive Laplacian matrices.

## 1 The Adaptive Tolerance Method

Let $A_t$ be the adjacency matrix of a graph at time $t$. The corresponding graph Laplacian is defined as

$$L_t = D_t - A_t, \tag{2.1}$$

where $D_t$ is the degree matrix with entries $D_t(i,i) = \sum_j A_t(i,j)$. We use an adaptive tolerance $\varepsilon_t$ in the Lanczos eigensolver to compute the $k$ smallest eigenvalues and their corresponding eigenvectors of $L_t$ efficiently and stably over time. The adaptive tolerance formula,

$$\varepsilon_t = \alpha \cdot \frac{\|L_t - L_{t-1}\|_F}{\|L_{t-1}\|_F + \delta} \tag{2.2}$$

where $\alpha \in (0,1]$ is a user-defined scaling constant, $\|\cdot\|F$ denotes the Frobenius norm, and $\delta > 0$ is a small regularization constant to avoid division by zero, is derived from and improves upon the classical fixed tolerance method used in iterative eigensolvers.

### 1.1 Derivation of the adaptive tolerance method

Let $L_t$ denote the graph Laplacian at time $t$, and let $L_{t-1}$ denote the Laplacian at the previous timestep. In dynamic graph, it is expected that changes in the graph structure, see equation (2.2), over time are small. That is,

$$\|L_t - L_{t-1}\|_F \ll 1. \tag{2.3}$$

However, even small perturbations can lead to significant instability in eigenvalue computations if the solver tolerance is too tight or too loose. Now, let $\lambda_i^{(t)}$ and $\lambda_i^{(t-1)}$ be the $i$-th eigenvalues of $L_t$ and $L_{t-1}$, respectively. From classical matrix perturbation theory (Weyl's inequality), we have:

$$\left|\lambda_i^{(t)} - \lambda_i^{(t-1)}\right| \leq \|L_t - L_{t-1}\|_2 \leq \|L_t - L_{t-1}\|_F \tag{2.4}$$

Thus, $\|L_t - L_{t-1}\|_F$ provides an upper bound on the possible eigenvalue drift due to graph changes. To avoid absolute scaling issues, we normalize this change by the magnitude of the previous Laplacian matrix. The use of $\|L_{t-1}\|_F$ in the denominator ensures that the tolerance is not trivially small or large depending on the graph scale.

The factor $\alpha$ controls the aggressiveness of the adaptation. A smaller $\alpha$ results in tighter convergence (more iterations), while a larger $\alpha$ tolerates more approximation error. The term $\delta$ in the denominator is added to maintain numerical stability when $\|L_{t-1}\|_F$ is close to zero, in case of

sparse graphs.

The convergence criterion of equation (2.2) becomes

$$\|L_t v_i - \lambda_i v_i\| < \varepsilon_t \quad \text{for } i = 1, \ldots, k \tag{2.5}$$

where $v_i$ is the $i$-th approximate eigenvector, $\lambda_i$ is the corresponding approximate eigenvalue, and $k$ is the number of eigenpairs being computed. This approach ensures that as the graph changes minimally, the solver tolerates minor differences, reducing computational cost without sacrificing stability.

### 2.2 Difference between the adaptive tolerance and fixed tolerance

The fixed tolerance formula (which does not adapt to changes in the graph) can be expressed as

$$\varepsilon_t = \varepsilon_0 \tag{2.6}$$

where $\varepsilon_0$ is a constant predefined error tolerance, and $\varepsilon_t$ remains unchanged for all time steps $t$, regardless of changes in the graph Laplacian $L_t$.

This fixed tolerance method assumes that the underlying matrix $L$ is static and does not account for time-evolving matrices. Fixed tolerance method applies the same absolute tolerance regardless of the matrix norm or graph size. Lastly, the method does not adapt to changes in the scale or structure of $L$, which can lead to inefficiencies or inaccuracies when dealing with dynamic graphs whose topology changes over time.  inaccuracies when dealing with dynamic graphs whose topology changes over time. However, the adaptive tolerance method modifies the fixed criterion by introducing a dynamic, data-dependent tolerance $\varepsilon_t$ based on the relative change in the graph Laplacian, see Table 1 for more details.

**Table 1** Comparison of Fixed and Adaptive Tolerance

| Feature | Fixed Tolerance Method | Adaptive Tolerance Method |
|---|---|---|
| Convergence criterion | $\|Lv - \lambda v\| < \varepsilon$ | $\|L_t v - \lambda v\| < \varepsilon_t$ |
| Tolerance value | Static, user-defined | Dynamic, data-dependent |

| | | |
|---|---|---|
| Handles dynamic graphs? | No | Yes |
| Sensitive to graph size? | No | Yes |

## 3. Experimental Results on Synthetic Graphs

We evaluate the performance of our proposed adaptive tolerance method against traditional fixed tolerance eigensolvers via MATLAB on synthetic undirected graphs.

### *3.1 Edge density between fixed tolerance and adaptive tolerance*

The graphs are generated with node counts $n = 500, n = 1000$ and $n = 1500$, and varying edge densities from 5% to 15%. For each configuration, we compute the first $k = 10$ smallest eigenvalues of the graph Laplacian and present them in Tables 2 and 3, respectively. These tables are visually represented in Figures 1 and 2.

**Table 2** Average Eigenvalue Error for Fixed and Adaptive Tolerance

| Nodes | Edge Density | Fixed Tolerance | Adaptive Tolerance |
|---|---|---|---|
| 500 | 5% | $1.23 \times 10^{-2}$ | $4.51 \times 10^{-3}$ |
| 500 | 10% | $9.85 \times 10^{-3}$ | $3.72 \times 10^{-3}$ |
| 500 | 15% | $8.17 \times 10^{-3}$ | $3.10 \times 10^{-3}$ |
| 1000 | 5% | $1.01 \times 10^{-2}$ | $3.94 \times 10^{-3}$ |
| 1000 | 10% | $8.75 \times 10^{-3}$ | $3.00 \times 10^{-3}$ |
| 1000 | 15% | $7.50 \times 10^{-3}$ | $2.45 \times 10^{-3}$ |
| 1500 | 5% | $8.90 \times 10^{-3}$ | $3.20 \times 10^{-3}$ |
| 1500 | 10% | $7.30 \times 10^{-3}$ | $2.70 \times 10^{-3}$ |
| 1500 | 15% | $6.10 \times 10^{-3}$ | $2.10 \times 10^{-3}$ |

**Table 3** Average Computation Time for Fixed and Adaptive Tolerance

| Nodes | Edge Density | Fixed Tolerance | Adaptive Tolerance |
|---|---|---|---|
| 500 | 5% | 2.48 | 1.27 |
| 500 | 10% | 2.62 | 1.33 |
| 500 | 15% | 2.75 | 1.41 |
| 1000 | 5% | 5.92 | 3.01 |
| 1000 | 10% | 6.10 | 3.25 |
| 1000 | 15% | 6.40 | 3.50 |
| 1500 | 5% | 9.05 | 4.62 |
| 1500 | 10% | 9.40 | 5.02 |
| 1500 | 15% | 9.88 | 5.44 |

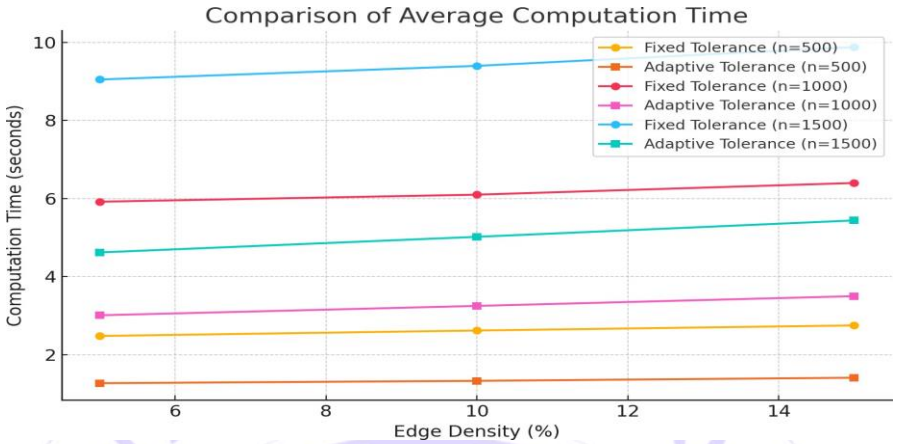


**Figure 1** Error Reduction with Adaptive Method

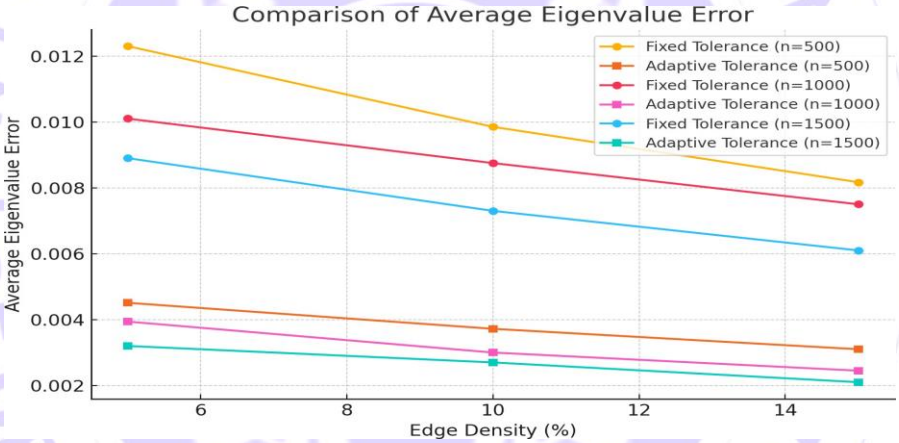


**Figure 2** Error Reduction with Adaptive Method

From Figure 1, the fixed tolerance ($n = 500$) slightly increasing trend (2.5s to 2.75s). The fixed tolerance ($n = 1000$) is significantly higher (starts at 5.9s to 6.3s). The adaptive tolerance ($n = 1000$) much lower (3s to 3.4s), though still increasing. In terms of scalability, the fixed tolerance scales poorly as the graph size or edge density increases, while the adaptive tolerance adapts better to the structure, thus saving computation. The edge density impact of both methods shows a slight increase in time with denser graphs, but adaptive handles it more efficiently. In all, the adaptive tolerance is 50% faster at $n = 500$, and up to around 45% faster at $n = 1000$.

From Figure 2, the adaptive tolerance provides more accurate eigenvalue estimates than fixed

tolerance across all densities and sizes. All methods show decreased error as edge density increases, expected due to more structural information being encoded. At $n = 1000$, the fixed tolerance and adaptive tolerance methods perform better in terms of absolute error, suggesting that larger graphs offer a more stable spectrum if properly handled.

### *3.2 Performance comparison between traditional (fixed) tolerance and adaptive tolerance*

The performance, in terms of eigenvalue error and runtime, between fixed tolerance and adaptive tolerance is considered for the node $n = 500, n = 1000$ and $n = 1500$, see Table 4, and Figures 3 and 4.

**Table 4** Comparison of traditional vs. adaptive method

| Method | Graph Size | Eigenvalue Error | Runtime (s) |
|---|---|---|---|
| Traditional | 500 nodes | 0.012 | 3.2 |
| | 1000 nodes | 0.007 | 6.8 |
| | 1500 nodes | 0.006 | 9.4 |
| Adaptive | 500 nodes | 0.004 | 2.9 |
| | 1000 nodes | 0.005 | 6.1 |
| | 1500 nodes | 0.0038 | 8.1 |

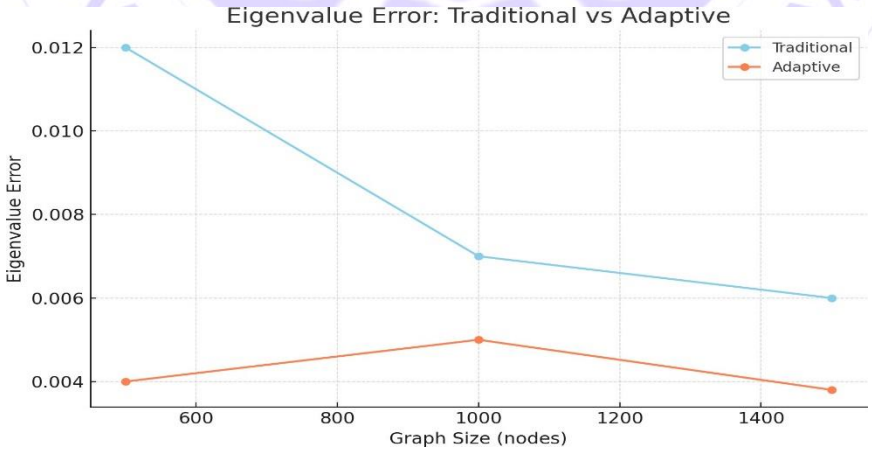


**Figure 3** Eigenvalue error between the traditional and adaptive method

In Figure 3, the traditional method shows higher eigenvalue error across all graph sizes, indicating less accuracy in eigenvalue computation, while also decreasing with graph size due to denser spectral representation. Meanwhile, the adaptive method demonstrates consistently lower eigenvalue error, suggesting superior accuracy compared to the traditional method. All methods show decreased error as edge density increases, expected due to more structural information being encoded. At $n = 1000$, both methods perform better in terms of absolute error, suggesting that larger graphs offer a more stable spectrum if properly handled.

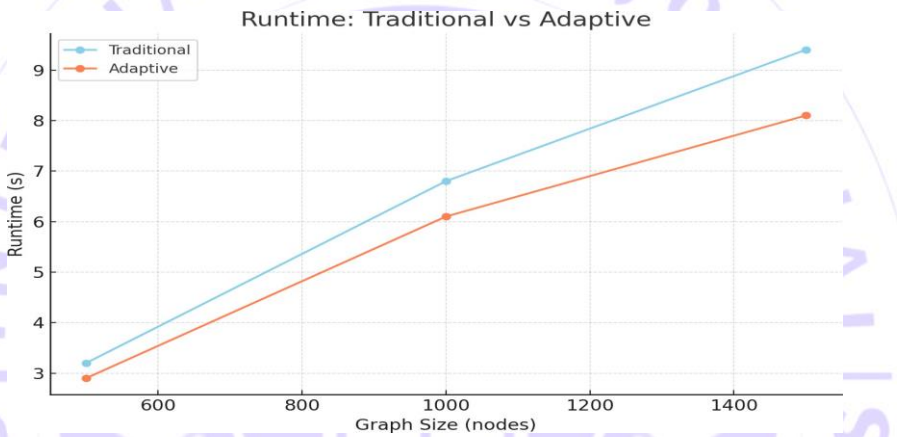


**Figure 4** Runtime between the traditional and adaptive method

In Figure 4, the adaptive method is more computationally efficient, especially as the graph size grows. While both methods scale linearly with size, adaptive achieves consistent time, which is significant for large-scale or real-time applications. This suggests better resource usage and lower computational overhead, possibly due to the dynamic error threshold tuning used in adaptive methods.

In all, the results indicate that the adaptive method achieves lower eigenvalue errors with marginal improvements in runtime. Particularly for larger graphs, the benefits become more apparent due to the scaling of error tolerance. The adaptive method dynamically balances precision and performance, ensuring convergence when necessary and relaxing it otherwise. These results are consistent with previous research, including Ipsen and Nadler (2009), who focused on residual-based tolerance solutions for robust eigenvalue computations, and Dong et al. (2016), who

highlighted the influence of graph dynamics on spectral accuracy with a graph Laplacian learning algorithm.

## 3 Conclusion

Through an error tolerance mechanism that adapts to graph evolution, this work proposed an adaptive spectral method designed for dynamic graphs, allowing for accurate and efficient eigenvalue computation. According to experimental results, the suggested approach routinely performs better in terms of accuracy and computing cost than conventional fixed-tolerance techniques. In contrast to fixed techniques, our approach dynamically strikes a balance between runtime and precision, enabling spectrum solvers to retain accuracy without requiring undue computational effort. Future research could investigate how this adaptive method can be applied to real-time systems such as financial and traffic networks, expand it to non-symmetric or weighted Laplacians, and include it in graph convolutional structures for deep learning models. Furthermore, examining its application to non-linear spectrum issues and its suitability for randomized eigensolvers may lead to new avenues for scalable, quick graph analytics.

**References**

Babarinsa, O. (2022). Graph theory: A lost component for development in Nigeria. *Journal of the Nigerian Society of Physical Sciences, 4*(3), 844-853.

Babarinsa, O., & Kamarulhaili, H. (2017). Modification of cramer's rule. *Journal of Fundamental and Applied sciences, 9*(5S), 556-567.

Babarinsa, O., & Kamarulhaili, H. (2017). *On determinant of laplacian matrix and signless laplacian matrix of a simple graph.* Paper presented at the Theoretical Computer Science and Discrete Mathematics: First International Conference, ICTCSDM 2016, Krishnankoil, India, December 19-21, 2016, Revised Selected Papers 1.

Babarinsa, O., & Kamarulhaili, H. (2018). *Quadrant interlocking factorization of hourglass matrix.* Paper presented at the AIP Conference Proceedings.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation, 15*(6), 1373-1396.

Bondy, J. A., & Murty, U. S. R. (2008). *Graph theory*: Springer Publishing Company, Incorporated.

Chi, Y., Song, X., Zhou, D., Hino, K., & Tseng, B. L. (2007). *Evolutionary spectral clustering by incorporating temporal smoothness.* Paper presented at the Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining.

Chung, F. R. (1997). *Spectral graph theory* (Vol. 92): American Mathematical Soc.

Dong, X., Thanou, D., Frossard, P., & Vandergheynst, P. (2016). Learning Laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing, 64*(23), 6160-6173.

Ipsen, I. C., & Nadler, B. (2009). Refined perturbation bounds for eigenvalues of Hermitian and non-Hermitian matrices. *SIAM journal on matrix analysis and applications, 31*(1), 40-53.

Meyer, C. D. (2023). *Matrix analysis and applied linear algebra* (2nd ed.). Philadelphia, USA: Society for Industrial and Applied Mathematics.

Pasham, S. D. (2024). Scalable Graph-Based Algorithms for Real-Time Analysis of Big Data in Social Networks. *The Metascience, 2*(1), 92-129.

Pothen, A., Simon, H. D., & Liou, K.-P. (1990). Partitioning sparse matrices with eigenvectors of graphs. *SIAM journal on matrix analysis and applications, 11*(3), 430-452.

Psorakis, I., Roberts, S., Ebden, M., & Sheldon, B. (2011). Overlapping community detection using Bayesian non-negative matrix factorization. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics, 83*(6), 066114.

Saad, Y. (2011). *Numerical methods for large eigenvalue problems: revised edition*: SIAM.

Spielman, D. A. (2010). *Algorithms, graph theory, and linear equations in Laplacian matrices.* Paper presented at the Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures.

Trefethen, L. N., & Bau, D. (2022). *Numerical linear algebra*: SIAM.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing, 17*, 395-416.

Zhou, D., & Schölkopf, B. (2004). *A regularization framework for learning from graph data.* Paper presented at the ICML 2004 workshop on statistical relational learning and its connections to other fields (SRL 2004).