Proceedings of the 4th International Conference on Applied Physics and Mathematical Modeling
DOI: 10.54254/2753-8818/2026.37899

Iterative Matrix Methods and Convergence Acceleration in Large-Scale Network Analysis

Yutong Zhu

San Pasqual High School, Escondido, USA
yutzhu232@euhsd.org

Abstract. Iterative matrix methods are central to large-scale network analysis because many ranking and centrality problems reduce to dominant-eigenvector, fixed-point, or sparse linear-system computations. This review summarises how adjacency, transition, and Google matrices support such formulations and compares major iterative approaches, including the power method, Jacobi and Gauss-Seidel iterations, Krylov subspace methods, and local or incremental updates. The discussion focuses on convergence speed, computational cost, memory requirements, approximation error, and scalability. It also examines the main factors that affect convergence, including matrix sparsity, spectral separation, damping parameters, stopping criteria, and initialisation, together with acceleration strategies such as extrapolation, Anderson acceleration, preconditioning, randomised sketching, and localised recomputation. Recent studies show that no single solver is uniformly superior because performance depends on matrix structure, update patterns, and hardware constraints. Important limitations include slow convergence for difficult spectra, rapidly evolving networks, parameter sensitivity, and inconsistent evaluation protocols. Future research should develop adaptive, randomised, and learning-assisted methods that retain numerical reliability while improving efficiency on dynamic large-scale networks.

Keywords: iterative matrix methods, large-scale network analysis, convergence acceleration

1. Introduction

Large-scale network datasets have been at the forefront of research in web retrieval, social network analysis, personalised recommendation systems and scientific information analysis, and the number of nodes and edges in many networks continues to grow into the millions or billions [1, 2]. Large-scale network analysis is feasible because the relationships among the nodes in a network can be expressed as a matrix, and thus, a structural problem of a network can be converted into an algebraic problem of a matrix. The most typical network analysis problems in this sense are node ranking and centrality calculation; they can be reduced to a small number of linear-algebraic problems by computing the dominant eigenvalue, the corresponding dominant eigenvector, or solving a linear system [2, 3]. Direct matrix factorisation of very large real-world networks, such as full eigenvalue decomposition or lower-upper (LU) decomposition, is too expensive in terms of both time and memory. Real-world networks are generally sparse, and computing or storing a dense node-proximity matrix requires computational time and storage that grow quadratically with the number

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2753-8818/2026.37899

of nodes; thus, when a network contains around ten million nodes, iterative and approximate methods that take advantage of sparsity and, in some cases, local or sublinear computation are used [1]. Therefore, iterative methods that approximate the target solution by carrying out multiple sparse matrix-vector products instead of explicit factorisation have become essential tools for large-scale network analysis due to their relatively low computational and memory consumption.

A large number of iterative methods have been developed for this environment. Traditionally, the starting point is the power method. It is the simplest solver for a dominant eigenvector and the foundation of PageRank and eigenvector centrality. Therefore, the literature has taken several routes: fixed-point iterations, Jacobi- and Gauss-Seidel-type splitting methods, Krylov subspace methods, extrapolation and momentum-based acceleration, and local or partial-update schemes [1, 4-7]. Efficiency of the above methods is also influenced by other factors, such as the sparsity of the matrix, the spectral gap, the damping parameter, the stopping criterion and the initial vector; these will be examined later. Recently, the three main directions of research by scholars have been: local and sublinear algorithms that update only the affected part of the solution [4]; learning-augmented approaches that combine neural networks with classical iteration [8]; and randomised or sketching-based solvers to reduce computational cost [9, 10]. PageRank is a representative case of how the same ranking problem can be solved using different iterative methods in the survey [1]. The primary focus is on the principles, convergence properties and applications of the methods rather than on search-engine implementation.

This paper focuses on iterative matrix methods and convergence acceleration for large-scale network analysis. The aim is to present how network matrices are converted into iteratively solvable problems and to compare the main families of these methods using five criteria: convergence speed, computational cost, memory requirement, approximation error and scalability. The following are the other parts of this paper. Section 2 presents the matrix representation and spectral basis of network analysis. Section 3 introduces some representative iterative matrix methods. Section 4 analyses convergence factors and acceleration strategies. Section 5 is about the shortcomings and future prospects; Section 6 concludes this paper.

2. Matrix representations and spectral foundations of network analysis

A network can be modelled as a graph consisting of a set of vertices and a set of edges, and may be either directed or undirected, weighted or unweighted. Real-world networks are generally sparse; therefore, the number of edges is relatively small compared to the square of the number of nodes, a property exploited by the iterative methods discussed in this survey. To perform algebraic analysis of such a network, its adjacency matrix is first constructed, with an entry of 1 if there is an edge from node j to node i and 0 otherwise. The adjacency matrix alone treats all existing edges equally and does not distinguish edge importance. A diagonal matrix records the out-degree of each node, and each column of the adjacency matrix is normalised by the corresponding out-degree to obtain the transition matrix. When every node has at least one outgoing edge, each column of the transition matrix is non-negative and sums to one; thus, it is column-stochastic, and each entry is the probability that a random walk at one node moves to another node in one step. Dangling nodes, which have no outgoing edges, produce zero columns and therefore require separate treatment. In PageRank, dangling columns are replaced by a chosen probability distribution, and then a damping and teleportation step are employed to construct the column-stochastic Google matrix.

The chain of the above matrices converges to a single question: given that importance is propagated recursively along edges, which vector is invariant to this propagation? A node is important if it is referred to by other important nodes. In eigenvector centrality, the centrality score

DOI: 10.54254/2753-8818/2026.37899

vector is an eigenvector of the adjacency matrix associated with its dominant eigenvalue, and the centrality vector is related to the spectral radius of the adjacency matrix. According to the Perron-Frobenius theorem for an irreducible non-negative adjacency matrix, there is a corresponding non-negative dominant eigenvector that can be regarded as a centrality score. A potential failure mode is localisation; in this case, the main eigenvector puts almost all its weight on a few high-degree hubs and fails to distinguish between the others. Previous research has examined localisation in eigenvector centrality [3].

PageRank is a related spectral problem that looks for a stationary vector that remains unchanged after multiplication by the Google matrix. Given the positive teleportation, the Google matrix is column-stochastic, irreducible, and aperiodic; thus, a unique stationary vector exists, and this vector is the dominant eigenvector associated with eigenvalue one. Therefore, the two formulations that often appear in this paper, node ranking as a fixed point and centrality as a dominant eigenvector, are closely related spectral formulations of different matrices.

3. Iterative matrix methods for large-scale networks

As shown in the previous section, the calculation of node rankings and centralities is equivalent to finding the dominant eigenvector or a fixed point of a large sparse matrix. As direct factorisation is not feasible at scale, these quantities are computed iteratively by means of repeated sparse matrix-vector products. The two most basic families of iterative methods introduced here are the power method and Krylov subspace methods, while a few other related methods are briefly mentioned.

3.1. The power method

The power method is an easy-to-use, general-purpose iterative solver for the dominant eigenvector that is employed in both PageRank and eigenvector centrality. Starting with an initial vector that has a non-zero component in the dominant eigenspace, the matrix is repeatedly applied to this vector and then normalised; under standard spectral conditions, the dominant component will eventually be retained. It is simple to implement in a large-scale network and has a low cost; only a sparse matrix–vector product and storage proportional to the number of nodes and edges are required per iteration, and no explicit matrix formation or factorisation is needed [1, 2].

A general disadvantage of the power method is that its speed of convergence asymptotically depends on the ratio of the magnitudes of subdominant and dominant eigenvalues. When these are close together, for example, when the damping factor of PageRank is close to 1, the non-dominant components decay slowly, and it may take a large number of iterations to converge. Only the single dominant eigenvector is returned, and therefore additional spectral information needed for some analyses cannot be obtained directly. These deficiencies motivate the more powerful methods discussed next and the acceleration strategies in Section 4.

3.2. Krylov subspace methods

Krylov subspace methods overcome the inefficiency of the power method by reusing information that the power method discards. The power method keeps only the latest iterate, but Krylov methods store the entire sequence of vectors generated by repeated multiplication and then find the best approximation in the subspace spanned by these vectors. Because the Krylov subspace contains the results of multiple matrix-vector multiplications, these methods can generally achieve a relatively accurate approximation in fewer iterations than the basic power method and are still applicable to

DOI: 10.54254/2753-8818/2026.37899

sparse matrix problems. Representative algorithms include Arnoldi iteration for general nonsymmetric matrices arising from directed networks and Lanczos iteration for symmetric matrices arising from undirected networks, where the adjacency relation between any two nodes is reciprocal [7].

They are relatively efficient but require more memory because the accumulated vectors need to be stored; therefore, this memory cost is often controlled in practice by periodically restarting the process. Krylov methods are also more versatile than the power method; they can obtain several eigenvalues and eigenvectors simultaneously, which is convenient for spectral analysis and for solving problems involving matrix functions, and they can be extended to solve the linear systems that arise in the fixed-point formulation of PageRank. Recently, some research has continued to develop this family of methods, such as employing randomised Krylov variants for large nonsymmetric systems [9] and training neural operators to predict effective subspaces to reduce the number of iterations [8].

3.3. Additional iterative methods

Other ways also build upon these two foundations. Classical stationary iterations, such as Jacobi and Gauss-Seidel, can be applied to the linear-system formulation of PageRank. When the relevant convergence conditions are met, Gauss-Seidel may require fewer iterations because it uses the newly updated values in each iteration. Local and incremental methods can avoid recalculating the entire solution by updating only the nodes affected by a query or by changes in the graph for personalised ranking and dynamic networks [1, 4]. These ways will be mentioned again in the discussion of acceleration and dynamic conditions later on.

4. Convergence analysis and acceleration strategies in network applications

4.1. Factors controlling convergence

The speed of convergence and stability for iterative methods applied to network matrices are affected by many factors, including the spectral gap between the dominant and subdominant eigenvalues, the damping factor in PageRank, the conditioning of the matrix, the choice of the initial vector, and the stopping tolerance. For PageRank, setting the damping factor closer to 1 gives more weight to the structure of links but can slow down the convergence of power iteration [1]. Gauss-Seidel may have fewer iterations than the Jacobi method for some linear systems due to its use of recently updated values, but it is not always better. Similarly, the convergence of the power and inverse power methods depends on the separation of the eigenvalues involved in each method. A larger relevant spectral separation generally leads to faster convergence, whereas a smaller separation can slow convergence. In short, all the above observations show that the speed of convergence is not an intrinsic property of the method, but rather a consequence of the interaction among parameter selection, iteration scheme, and spectral properties of the network.

4.2. Acceleration strategies

Based on the above, some strategies can reduce the number of iterations or the cost per iteration while preserving accuracy. Extrapolation and momentum-based methods combine several past iterates to eliminate the slowest-decaying error component directly, and are effective for accelerating fixed-point formulations such as multilinear PageRank [5]. Anderson acceleration is a related but

DOI: 10.54254/2753-8818/2026.37899

more general method that forms a least-squares combination of recent residuals instead of a fixed extrapolation rule, and it has been used to accelerate fixed-point iterations for multilinear PageRank [6]. A second family of strategies modifies the search space or improves the conditioning of the problem rather than directly accelerating a fixed iteration. Krylov subspace methods expand the search space beyond a single iterate [7, 9], and explicit preconditioning constructs a transformed system that is easier to solve [10]. A third approach is to reduce computational cost by using randomised Krylov or sketching-based methods [9, 10]. When only a part of the network changes, incremental methods can update only the affected region instead of recomputing the entire ranking from scratch, reducing unnecessary work in the unchanged parts of the graph [1, 4].

4.3. How to evaluate a method

In order to determine which method is more suitable, the three related ideas of the theoretical convergence rate, the actual running time, and the quality of approximation need to be distinguished. Gauss-Seidel may have fewer iterations than Jacobi for some problems, but Jacobi is more suitable for parallelisation, as the updates at each iteration can be carried out independently. Therefore, a method with a slower sequential convergence rate can still be used for parallel computing. The convergence speeds of the power and inverse power methods can differ because they are applied to different parts of the spectrum. Neither is intrinsically better; therefore, the appropriate choice depends on the spectral target. A practical evaluation method for PageRank-type problems should start with the construction of the matrix and its sparsity, select an iterative scheme suitable for the expected spectral structure, monitor the approximation error relative to a stopping tolerance instead of a fixed number of iterations, and then determine whether an acceleration strategy is worthwhile. A method that halves the number of iterations but doubles the cost per iteration offers no practical advantage; therefore, runtime and approximation quality should be considered along with convergence speed, rather than substituted for it.

5. Limitations and prospects

Iterative matrix methods for large networks have several limitations. Real networks are constantly changing, and there may be noise or missing links; thus, a solution computed once can quickly become obsolete [1]. A scheme optimised for a single network snapshot may no longer be effective after a change in link structure, and incremental methods only partially solve this problem by requiring the identification and propagation of updates in the affected area. Small spectral gaps lead to slow convergence of classical methods [3]. Different damping factors and stopping criteria can produce different rankings; thus, no universally agreed setting exists [1]. As these selections can affect both the convergence and the numerical results, comparisons among the studies in different environments may be due to variations in parameters or different methods. Theoretical advantages do not always translate into practical efficiency, and differences in datasets and evaluation metrics also make direct comparisons difficult.

Several approaches can address these limitations. Incremental and local updates for large or dynamic networks reduce recomputation [1, 4]. The effectiveness of such methods under continuous high-frequency changes remains unclear. Anderson acceleration can improve convergence [6], and randomised and sketched iterative methods enhance scalability [9, 10]. Combining iterative matrix methods with graph machine learning can further reduce the computational cost through learned solvers or neural-guided iteration [8]. However, such learned methods depend on training data that adequately represent the target problem.

DOI: 10.54254/2753-8818/2026.37899

6. Conclusion

This paper reviews iterative matrix methods and their convergence acceleration in large-scale network analysis. First, in terms of adjacency, transition and Google matrices, node ranking and centrality can be formulated as dominant-eigenvector or fixed-point calculations for network structure. Next, some basic iterative solvers are introduced: the power method, classical fixed-point splitting schemes such as Jacobi and Gauss-Seidel, Krylov subspace methods, and local or incremental update schemes, and the trade-offs among iteration cost, memory requirements and applicability for each are analysed. As shown in the above analysis, the spectral gap, damping factor and selection of iteration scheme also affect convergence. Acceleration strategies include Anderson acceleration, preconditioning and incremental computation; they can reduce either the number of iterations or the cost per step without sacrificing accuracy. Among these ways, none is suitable for all situations. The power method is relatively simple and has a small memory footprint, but Krylov methods can converge in fewer iterations at the cost of higher memory consumption. Local or incremental methods can be used if only a part of a large network needs to be updated. Traditional approaches have problems with data and parameter updates, inconsistent evaluation standards, and the difficulty of large-scale computation. Future evaluations should report data on convergence speed, execution time and memory usage at various accuracy levels. The above reporting conditions are necessary for the reproducibility of large-scale network studies. In the future, research should investigate adaptive, randomised and localised iterations and integrate them with graph machine learning. In short, linear-algebraic methods provide the basis for large-scale data-science analysis.

References

[1] Yang, M., Wang, H., Wei, Z., Wang, S., & Wen, J.-R. (2024). Efficient algorithms for personalized PageRank computation: A survey. *IEEE Transactions on Knowledge and Data Engineering, 36*(9), 4582–4602. https://doi.org/10.1109/TKDE.2024.3376000

[2] Chen, D., Chen, J., Zhang, X., Jia, Q., Liu, X., Sun, Y., Lü, L., & Yu, W. (2025). Critical nodes identification in complex networks: A survey. *Complex Engineering Systems, 5*, Article 11. https://doi.org/10.20517/ces.2025.34

[3] Sharkey, K. J. (2019). Localization of eigenvector centrality in networks with a cut vertex. *Physical Review E, 99*(1), 012315. https://doi.org/10.1103/PhysRevE.99.012315

[4] Bressan, M., Peserico, E., & Pretto, L. (2023). Sublinear algorithms for local graph-centrality estimation. *SIAM Journal on Computing, 52*(4), 968–1008. https://doi.org/10.1137/19M1266976

[5] Bentbib, A. H., Boubekraoui, M., & Jbilou, K. (2025). Extrapolation methods for multilinear PageRank. *Numerical Algorithms, 98*(2), 1013–1043. https://doi.org/10.1007/s11075-024-01823-x

[6] Lai, F., Li, W., Peng, X., & Chen, Y. (2023). Anderson accelerated fixed-point iteration for multilinear PageRank. *Numerical Linear Algebra with Applications, 30*(5), e2499. https://doi.org/10.1002/nla.2499

[7] Amsel, N., Chen, T., Greenbaum, A., Musco, C., & Musco, C. (2024). Nearly optimal approximation of matrix functions by the Lanczos method. In *Advances in Neural Information Processing Systems (Vol. 37)*, pp. 139823–139853. https://doi.org/10.52202/079017-4437

[8] Luo, J., Wang, J., Wang, H., Dong, H., Geng, Z., Chen, H., & Kuang, Y. (2024). Neural Krylov iteration for accelerating linear system solving. In *Advances in Neural Information Processing Systems (Vol. 37)*, pp. 128636–128667. https://doi.org/10.52202/079017-4086

[9] Meurant, G. (2025). A randomized Q-OR Krylov subspace method for solving nonsymmetric linear systems. *Mathematics, 13*(12), 1953. https://doi.org/10.3390/math13121953

[10] Dereziński, M., Musco, C., & Yang, J. (2025). Faster linear systems and matrix norm approximation via multi-level sketched preconditioning. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1972–2004. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611978322.62