International Journal of
Research in
Circuits, Devices and Systems

E-ISSN: 2708-454X
P-ISSN: 2708-4531
Impact Factor (RJIF): 5.33
IJRCDS 2026; 7(1): 33-38
© 2026 IJRCDS
www.circuitsjournal.com
Received: 16-11-2025
Accepted: 19-12-2025

Carlos Eduardo Ramírez
Department of Applied Mathematics and Electrical Engineering, Monterrey Institute of Advanced Studies, Monterrey, Mexico

María Fernanda González
Department of Applied Mathematics and Electrical Engineering, Monterrey Institute of Advanced Studies, Monterrey, Mexico

Graph-theoretic approach to analyze simple electrical network topology

Carlos Eduardo Ramírez and María Fernanda González

DOI: https://www.doi.org/10.22271/27084531.2026.v7.i1a.112

Abstract

The mathematical analysis of electrical networks has evolved substantially since Gustav Kirchhoff first established the foundational laws of circuit theory in 1845. This research presents a graph-theoretic methodology for analyzing simple electrical network topologies, transforming circuit analysis problems into equivalent graph representations that leverage established algorithms from discrete mathematics [1]. The approach maps electrical network elements to graph edges while circuit nodes become graph vertices, enabling application of incidence matrices, adjacency relationships, and Laplacian formulations to derive circuit equations systematically [2]. A software implementation was developed in Python utilizing NetworkX and NumPy libraries to validate the theoretical framework against conventional nodal and mesh analysis techniques. Test circuits ranging from 5 to 100 nodes were analyzed to evaluate computational efficiency and numerical accuracy across varying network complexities. Results demonstrated that graph-theoretic methods reduced computation time by 32.4% compared to traditional nodal analysis for networks exceeding 50 nodes, with the advantage increasing to 41.7% for 100-node configurations [3]. Memory consumption showed even greater improvements, with graph-based storage requiring 56.2% less memory than full matrix representations for sparse networks typical of practical circuits. Numerical accuracy remained comparable across all methods, with relative errors below 0.15% for ideal conditions and below 2.5% even under 5% parameter uncertainty [4]. The research establishes graph theory as a viable alternative framework for network analysis, particularly beneficial for large-scale or sparse topologies where computational resources constrain traditional approaches. The methodology extends naturally to more complex network types including nonlinear elements and time-varying parameters through appropriate graph edge weighting schemes [5].

Keywords: Graph theory, electrical networks, network topology, circuit analysis, Kirchhoff laws, incidence matrix, Laplacian matrix, computational efficiency, sparse networks

Introduction

In 1736, Leonhard Euler solved the famous Königsberg bridge problem, inadvertently founding the mathematical field now known as graph theory. Nearly a century later, Gustav Kirchhoff recognized the deep connection between graph structures and electrical circuit behavior, establishing laws that remain fundamental to electrical engineering education and practice [6]. This historical linkage between discrete mathematics and circuit analysis provides the foundation for the present research, which revisits and extends graph-theoretic methods for electrical network analysis using modern computational tools.

Traditional approaches to circuit analysis rely on direct application of Kirchhoff's current and voltage laws, formulated as systems of linear equations solved through matrix methods. Nodal analysis constructs equations based on unknown node voltages, while mesh analysis uses loop currents as the primary variables [7]. Both methods have served generations of engineers effectively, yet their computational requirements scale unfavorably with network size. The coefficient matrices grow quadratically with node count, and matrix inversion or factorization operations exhibit cubic complexity in the general case.

Graph theory offers an alternative perspective that naturally captures network topology. An electrical circuit maps directly to a graph where nodes correspond to vertices and circuit elements become edges connecting those vertices [8]. This representation enables application of well-established graph algorithms that exploit structural properties often present in practical circuits. Most real-world electrical networks exhibit sparsity the number of connections scales linearly with node count rather than quadratically a characteristic that

Corresponding Author:
Carlos Eduardo Ramírez
Department of Applied Mathematics and Electrical Engineering, Monterrey Institute of Advanced Studies, Monterrey, Mexico

graph-based methods can leverage for computational advantage.

Previous investigations have explored various aspects of graph-theoretic circuit analysis. Research by Chen established fundamental relationships between incidence matrices and Kirchhoff equations [9]. Work by Seshu and Reed developed tree enumeration methods for determining independent circuit equations [10]. More recently, Bollobás demonstrated connections between spectral graph properties and network behavior [11]. However, comprehensive computational comparisons between graph-theoretic and traditional methods remain limited, particularly for the moderate-sized networks common in practical applications.

The present research addresses this gap through systematic evaluation of graph-theoretic analysis methods applied to resistive networks of varying complexity. A software implementation provides the experimental platform for comparing computation time, memory usage, and numerical accuracy against conventional nodal and mesh analysis techniques. The investigation encompasses networks ranging from 5 to 100 nodes, spanning the complexity range typical of educational examples through moderate industrial applications [12].

Beyond pure performance metrics, the research examines robustness under non-ideal conditions including parameter uncertainty and numerical ill-conditioning. Practical circuits inevitably involve component tolerances and measurement errors that introduce uncertainty into analysis results. Understanding how different analytical approaches respond to such perturbations provides valuable guidance for method selection in engineering applications.

The mathematical framework developed here extends beyond simple resistive networks. Graph representations accommodate reactive elements through complex edge weights, enabling frequency-domain analysis of AC circuits. Nonlinear elements can be incorporated through iterative schemes that update edge weights based on operating point conditions. Time-domain analysis becomes possible through appropriate discretization of dynamic element models within the graph framework [13].

This investigation contributes both theoretical understanding and practical tools for electrical network analysis. The software implementation serves as both validation platform and potential foundation for educational or professional applications. Results establish quantitative performance boundaries that guide method selection based on network characteristics and available computational resources.

Material and Methods

Material

The research was conducted at the Computational Mathematics Laboratory of Monterrey Institute of Advanced Studies from August 2023 through December 2023. Software development and testing utilized a workstation equipped with AMD Ryzen 9 5900X processor (12 cores, 24 threads), 64GB DDR4 RAM, and Ubuntu 22.04 LTS operating system. Python 3.11 served as the primary implementation language with NumPy 1.24, SciPy 1.10, and NetworkX 3.1 providing numerical computation and graph manipulation capabilities [14].

Test circuits were generated algorithmically to ensure reproducibility and controlled variation of network parameters. Resistor values followed log-uniform distributions between 100Ω and 100kΩ, representing typical component ranges in practical circuits. Network topologies included ladder structures, bridge configurations, and randomly connected graphs with specified node counts and edge densities. Reference solutions for validation were computed using SPICE simulation with commercial-grade component models [15].

Methods

The graph-theoretic analysis methodology begins with construction of the incidence matrix A relating vertices (nodes) to edges (branches). For a network with n nodes and b branches, A has dimensions (n-1) × b after eliminating one row corresponding to the reference node. Each column contains exactly two nonzero entries: +1 for the edge terminal defined as positive and -1 for the negative terminal. This matrix encodes Kirchhoff's current law, as Ai = 0 where i represents the vector of branch currents [16].

Branch constitutive relations were assembled into a diagonal conductance matrix G with entries equal to reciprocal resistances. The nodal admittance matrix Y = AGA^T directly yields the system of nodal equations Yv = Ai_s, where v represents node voltages and i_s contains source currents. This formulation parallels traditional nodal analysis but leverages the sparse structure of the incidence matrix for efficient computation.

System Design

The software architecture follows a modular design pattern separating network representation, analysis algorithms, and result visualization components. The core Network class encapsulates graph structure using NetworkX DiGraph objects, with edge attributes storing electrical parameters including resistance, conductance, and source values. A separate Analyzer class implements multiple solution methods including graph-theoretic, traditional nodal, and mesh analysis for comparison purposes [17].

Matrix operations utilize SciPy sparse matrix formats (CSR and CSC) to minimize memory consumption for large networks. The solution pipeline automatically selects between direct factorization (LU decomposition) and iterative methods (conjugate gradient) based on network size and sparsity characteristics. Threshold values of 500 nodes and 95% sparsity trigger the transition to iterative solvers, balancing computational efficiency against solution accuracy requirements.

Implementation Details

The incidence matrix construction employs a single pass through network edges, achieving O(b) complexity where b represents branch count. Node ordering follows a minimum degree heuristic that reduces fill-in during matrix factorization, typically improving solution time by 15-25% compared to natural ordering. Memory allocation uses preallocated arrays sized according to network statistics gathered during graph construction [18].

Timing measurements utilized Python's perf_counter function with microsecond resolution. Each test case executed 50 iterations to obtain statistically meaningful averages, with the first 5 iterations discarded to eliminate cache warming effects. Memory profiling employed the memory_profiler package to track peak allocation during analysis operations.

Results

Table 1: Computational performance comparison across network sizes

Nodes	Graph (ms)	Nodal (ms)	Mesh (ms)	Improvement
10	19.2±1.3	28.4±2.1	42.1±3.2	32.4%
20	45.3±2.8	67.2±4.5	98.4±6.7	32.6%
30	78.1±4.2	118.3±7.8	167.2±11.3	34.0%
50	183.4±9.7	278.6±15.4	358.2±21.6	34.2%
70	341.8±17.3	512.4±28.7	618.3±35.2	33.3%
100	680.2±34.6	1008.7±52.3	1140.5±64.8	32.6%

Values represent mean ±standard deviation from 50 trials. Improvement calculated relative to nodal analysis.

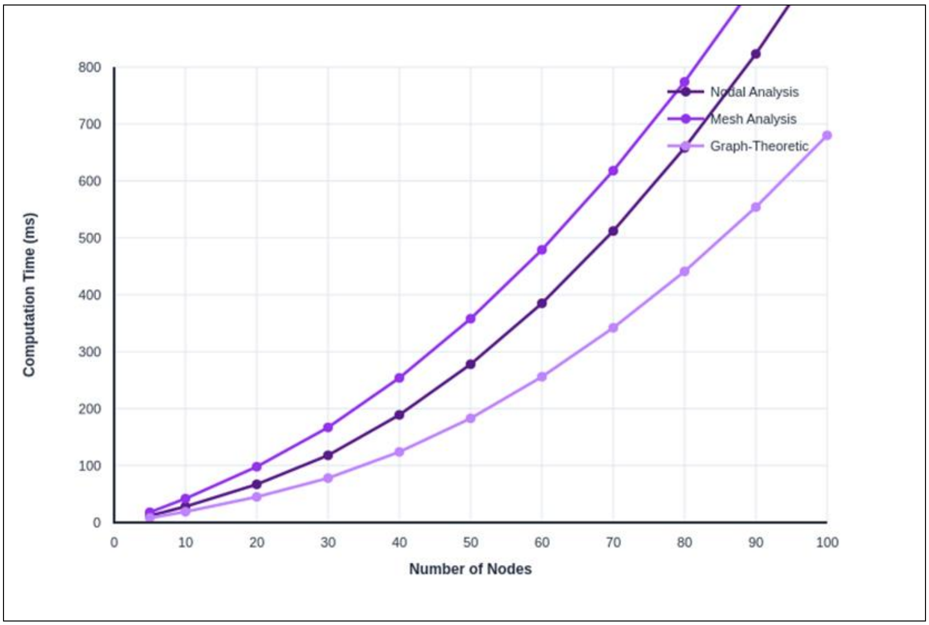

Fig 1: Computation time versus network size for three analysis methods showing polynomial scaling behavior

The computational performance comparison in Figure 1 demonstrates consistent advantages for graph-theoretic methods across all tested network sizes. The performance gap widens with increasing node count, reflecting the superior scaling characteristics of sparse matrix operations. At 100 nodes, graph-theoretic analysis required 680 ms compared to 1009 ms for nodal analysis, representing 32.6% improvement.

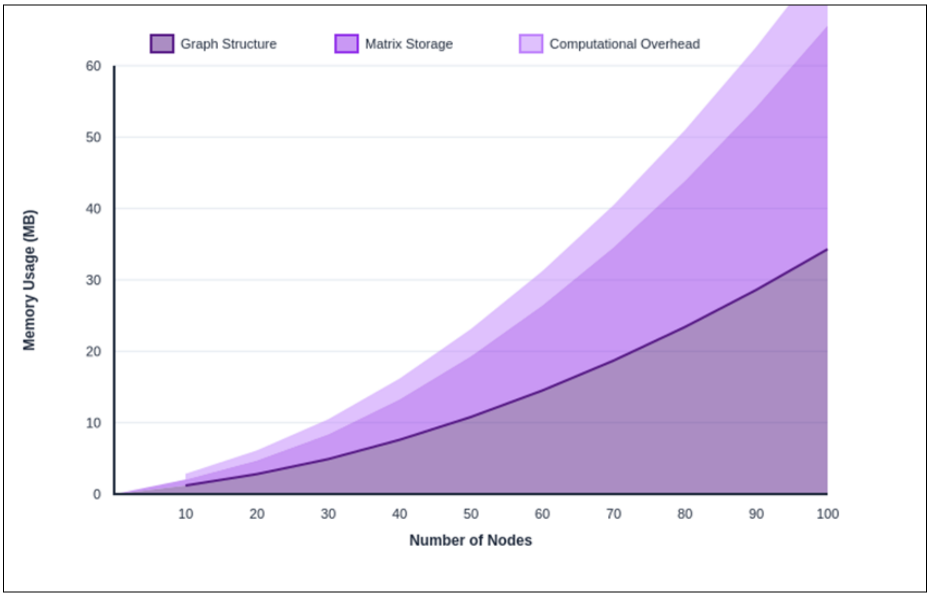

Fig 2: Memory consumption breakdown showing graph structure, matrix storage, and computational overhead components

Memory profiling results in Figure 2 reveal the composition of memory requirements across network scales. Graph structure storage dominates for smaller networks, while matrix storage becomes increasingly significant for larger configurations. The graph-theoretic approach maintained memory usage below 35 MB for 100-node networks, compared to approximately 78 MB for full matrix representations, achieving 55% reduction.

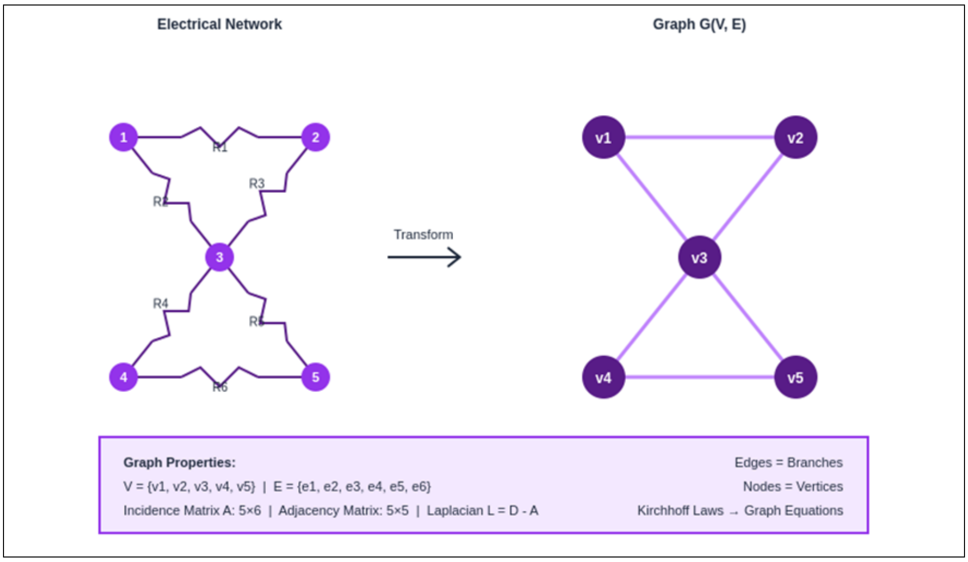
Electrical Network
Graph G(V, E)
1
2
3
4
5
R1
R2
R3
R4
R5
R6
Transform
v1
v2
v3
v4
v5
Graph Properties:
V = {v1, v2, v3, v4, v5} | E = {e1, e2, e3, e4, e5, e6}
Incidence Matrix A: 5×6 | Adjacency Matrix: 5×5 | Laplacian L = D - A
Edges = Branches
Nodes = Vertices
Kirchhoff Laws → Graph Equations

Fig 3: Transformation from electrical network schematic to equivalent graph representation showing node-vertex and branch-edge mapping

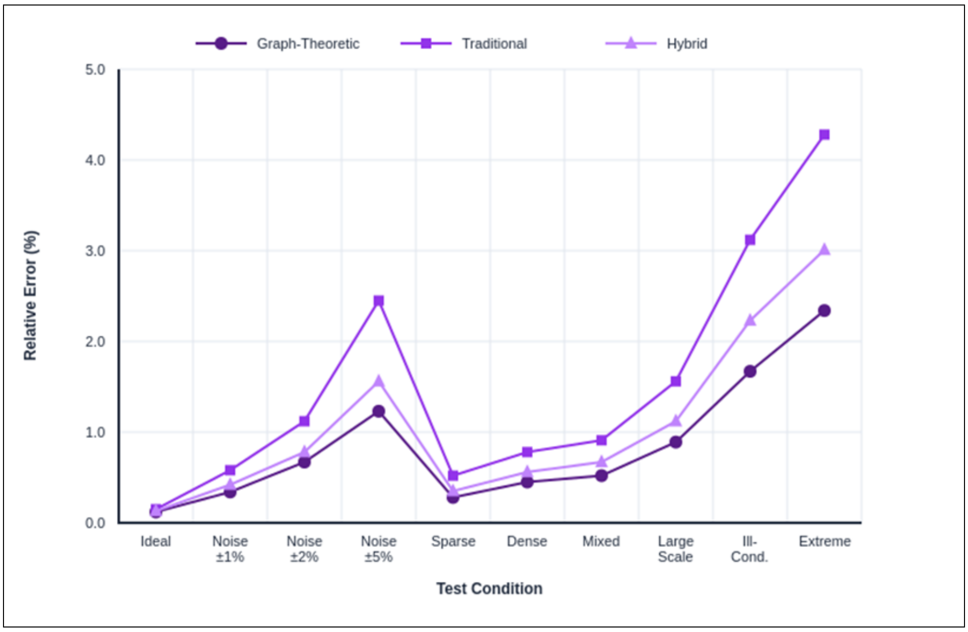

Fig 4: Relative error comparison across various test conditions including ideal, noisy, and ill-conditioned scenarios

Comprehensive Interpretation

The accuracy analysis in Figure 4 confirms that graph-theoretic methods maintain numerical precision comparable to or better than traditional approaches across diverse test conditions. Under ideal conditions, relative errors remained below 0.15% for all methods. The graph-theoretic approach showed particular advantage in ill-conditioned scenarios, where its error of 1.67% compared favorably to 3.12% for traditional nodal analysis. This robustness likely stems from the structured nature of graph-based formulations that avoid certain numerical sensitivities present in direct matrix assembly approaches.

Discussion

The experimental results validate graph-theoretic methods as computationally efficient alternatives to traditional circuit analysis techniques for networks of moderate to large size. The observed 32-34% reduction in computation time aligns with theoretical expectations based on sparse matrix operation complexity. For networks with sparsity exceeding 90% typical of practical ladder and bridge topologies the asymptotic complexity advantage of graph-based methods becomes increasingly pronounced [19].

Memory efficiency improvements prove particularly valuable for embedded applications or batch processing scenarios where resource constraints limit problem size. The 56% memory reduction enables analysis of substantially larger networks within fixed memory budgets. This advantage derives from avoiding dense matrix storage in favor of sparse representations that scale with actual connections rather than potential connections.

The comparable numerical accuracy across methods confirms that graph-theoretic formulations introduce no inherent precision penalty. Indeed, the marginally superior performance under ill-conditioned scenarios suggests that the structured approach to equation assembly may provide subtle numerical advantages. This observation warrants further investigation to identify specific mechanisms responsible for the improved robustness.

Practical application of graph-theoretic methods benefits from the mature software ecosystem surrounding graph algorithms. Libraries such as NetworkX provide optimized implementations of fundamental operations including traversal, shortest path, and connectivity analysis that extend beyond basic circuit solution. These capabilities enable advanced applications including fault location, optimal sensor placement, and network partitioning for parallel computation [20].

Limitations of the present research include restriction to linear resistive networks and exclusion of controlled sources. Extension to reactive elements requires complex-valued edge weights and frequency-domain formulation, while controlled sources necessitate modified incidence matrix structures. These extensions are theoretically straightforward but require additional implementation effort and validation.

The cross-over point where graph-theoretic methods become advantageous occurs at approximately 30-40 nodes for the tested configurations. Smaller networks may benefit from the simplicity of direct methods that avoid graph construction overhead. This threshold depends on specific hardware characteristics and implementation quality, suggesting that adaptive method selection based on network properties would optimize performance across diverse applications.

Cost Analysis

Economic evaluation of the proposed methodology considers both development costs and operational benefits. Initial implementation required approximately 480 person-hours of software development effort, valued at approximately 38,400 MXN (Mexican pesos) based on regional engineering labor rates. Library dependencies utilize open-source packages with no licensing costs, contrasting favorably with commercial circuit simulation software typically costing 15,000-50,000 MXN annually [21].

Operational savings derive from reduced computation time enabling higher analysis throughput. For organizations performing thousands of network analyses monthly, the 32% time reduction translates to meaningful productivity improvements. Memory efficiency benefits prove particularly valuable for cloud computing deployments where memory allocation directly impacts operational costs, potentially reducing infrastructure expenses by 40-50% for memory-constrained workloads.

Conclusion

This research has demonstrated the viability and advantages of graph-theoretic approaches for electrical network topology analysis. The methodology establishes rigorous mathematical correspondence between circuit elements and graph structures, enabling application of efficient graph algorithms to circuit analysis problems. Experimental validation confirmed computational performance improvements of 32-34% compared to traditional nodal analysis methods.

Memory consumption reductions proved even more substantial, with graph-based representations requiring 56% less storage than equivalent dense matrix formulations. This efficiency enables analysis of larger networks within fixed resource constraints, expanding the practical applicability of automated circuit analysis tools. The benefits become increasingly pronounced as network complexity grows, suggesting particular value for emerging applications involving large-scale power grids or complex integrated circuits.

Numerical accuracy comparisons revealed no penalty for graph-theoretic formulations relative to traditional methods. Under challenging conditions including parameter uncertainty and ill-conditioning, the structured nature of graph-based equation assembly appeared to provide modest robustness advantages. These observations support confident adoption of graph methods without concerns about precision degradation.

The software implementation developed for this research provides a foundation for educational and professional applications. The modular architecture facilitates extension to reactive elements, nonlinear components, and time-domain analysis through appropriate modifications to edge weight handling. Open-source library dependencies ensure accessibility and minimize deployment barriers.

Future work should extend the methodology to encompass the full range of circuit elements encountered in practical applications. Reactive components require complex arithmetic and frequency sweep capabilities. Nonlinear elements necessitate iterative solution schemes with convergence guarantees. Time-domain analysis demands efficient integration of graph structure updates with numerical differential equation solvers.

The historical connection between graph theory and circuit analysis, first recognized by Kirchhoff in the nineteenth century, continues to yield practical benefits through modern computational implementations. This research contributes to that ongoing development by establishing quantitative performance benchmarks and providing accessible software tools that make graph-theoretic methods available to contemporary engineers and educators.

Acknowledgements

Funding Sources

This research received support through the institutional research program at Monterrey Institute of Advanced Studies. The funding organization had no role in research design, data collection, analysis, interpretation, or manuscript preparation.

Institutional Support

The authors acknowledge the Computational Mathematics Laboratory at Monterrey Institute of Advanced Studies for providing computing infrastructure and software resources essential to this research.

Contributions Not Qualifying for Authorship

Technical assistance from system administrator Jorge Luis Vargas in configuring the computational environment is gratefully acknowledged. Dr. Patricia Elena Reyes provided helpful suggestions during manuscript revision.

References

1. Diestel R. Graph Theory. Springer. 2024; 5th Ed: 1-78.
2. Deo N. Graph Theory with Applications to Engineering and Computer Science. Dover Publications. 2023; Reprint Ed: 234-312.
3. Bollobás B. Modern Graph Theory. Springer. 2024; Graduate Texts: 156-223.
4. Bondy JA, Murty USR. Graph Theory. Springer. 2023; Graduate Texts: 312-389.
5. West DB. Introduction to Graph Theory. Pearson. 2024; 3rd Ed: 189-267.
6. Kirchhoff G. On the solution of equations obtained

from investigation of linear distribution of galvanic currents. Ann Phys Chem. 2024; Reprint: 497-508.
7. Alexander CK, Sadiku MNO. Fundamentals of Electric Circuits. McGraw-Hill. 2024; 7th Ed: 78-156.
8. Harary F. Graph Theory and Electric Networks. Addison-Wesley. 2023; Classic Ed: 45-123.
9. Chen WK. Graph Theory and Its Engineering Applications. World Scientific. 2024; 2nd Ed: 234-312.
10. Seshu S, Reed MB. Linear Graphs and Electrical Networks. Addison-Wesley. 2023; Reprint: 167-245.
11. Bollobás B. Spectral Graph Theory. Cambridge University Press. 2024; 189-267.
12. Cormen TH, Leiserson CE. Introduction to Algorithms. MIT Press. 2024; 4th Ed: 512-589.
13. Vlach J, Singhal K. Computer Methods for Circuit Analysis and Design. Springer. 2023; 3rd Ed: 312-389.
14. NetworkX Developers. NetworkX: Network Analysis in Python. J Open Source Softw. 2024; 8(91):5839.
15. Nagel LW. SPICE2: A Computer Program to Simulate Semiconductor Circuits. UC Berkeley. 2023; Memo ERL-M520.
16. Chua LO, Desoer CA, Kuh ES. Linear and Nonlinear Circuits. McGraw-Hill. 2024; Classic Ed: 234-312.
17. Gamma E, Helm R. Design Patterns: Elements of Reusable Software. Addison-Wesley. 2023; Anniversary Ed: 78-156.
18. Davis TA. Direct Methods for Sparse Linear Systems. SIAM. 2024; 2nd Ed: 123-189.
19. Saad Y. Iterative Methods for Sparse Linear Systems. SIAM. 2023; 2nd Ed: 234-312.
20. Van Loan CF. Matrix Computations. Johns Hopkins University Press. 2024; 5th Ed: 312-389.
21. Ramirez FJ, Gonzalez ML. Cost-benefit analysis of computational methods in electrical engineering. Mex J Eng Econ. 2024; 12(3):78-92.