

Numerical Methods for Solving Large Sparse Linear Systems

Mohammed Farhan Ibrahim¹, Sahib Abdullah Hasan²

¹Department of Mathematics, Shahid Madani University, Tabriz, Iran.

²Department of Mathematics, Shahid Madani University, Tabriz, Iran.

Article Info

Article history:

Received Sept., 03, 2025

Revised Oct., 20, 2025

Accepted Nov., 25, 2025

Keywords:

Sparse matrices

Iterative methods

Preconditioning

Conjugate Gradient

Parallel computing

ABSTRACT

Often used in engineering, scientific computing, and optimization, the study investigates numerical approaches for managing big sparse linear systems. Regarding memory Many of whose entries are zeros, sparse matrices will benefit greatly in large systems using both computational speed and use. Resolving such systems, however, creates difficulties in memory usage and computational complexity, and numerical stability. The essay contrasts direct and iterative methods along with their respective advantages and disadvantages. Direct techniques such LU decomposition yield right results, for larger systems, however, they are rather hard. Iterative methods such as the Conjugate Gradient (CG) method become useful for badly conditioned system (when combined with preconditioning), and can be very beneficial when convergence is accelerated. This article describes the method of solving large sparse linear systems that exist in the real-world using parallelization and preconditioning in combination with iterative methods.

Corresponding Author:

Mohammed Farhan Ibrahim

Department of Mathematics, Shahid Madani University

East Azarbaijan Province, Tabriz, Iran

Email: nihadcenterpublications@gmail.com

1. INTRODUCTION

The need to solve very large sparse linear systems has increased most significantly in several areas, including optimization, scientific computing, and engineering. The parts that are zero, will provide storage efficiency and speed improvements. However, the same sparsity that could make Direct methods impractical for larger systems of equations means that there are usually, unreasonably expensive computational costs and memory overhead on classical direct methods like LU Decomposition. [1]

Methods, such as the Conjugate Gradient (CG) method, have become popular because they can efficiently alter an initial guess to resolve large sparse systems. To solve these constraints many times over cycles, means, on average, that these methods are useful to solving difficult systems in one-tenth of the time it would have taken direct methods, utilizing advances like preconditioning which influence and accelerate convergence rates, along with parallel computation to reducing the time to compute solutions. [2]

Direct vs iterative approaches vary significantly, therefore this paper reviews the major numerical methods for solving large sparse linear systems, while providing discussion of the advantages, disadvantages & implementation of direct and iterative methods. An overview of parallel processing and preconditioning is also presented, [[n[other new technologies can substantially boost the performance of these algorithms, and hence enable the effective management of largescale systems in a multitude of real world situations]]. The research is intended to provide an explicit understanding of the issues and solutions related to the problem of solvig large sparse linear systems, thus leading to more productive computational models in scientific computing, engineering, and optimization. [3]

2. METHOD

2.1. Problem Definition and Dataset

Optimization, engineering, and scientific computing are common research topics that focus on finding solutions to large, sparse linear systems. Sparse matrices, which are matrices where a majority of components are zeros, create the data set and present computational challenges in both memory and speed of solution.

2.2. Numerical Methods

This study looks at two main groups of numerical approaches: direct methods and iterative algorithms; both have particular applications and performance characteristics for sparse systems.

2.2.1 Direct Methods

By simplifying the linear systems, direct techniques like Gaussian elimination and LU decomposition solve them. While these techniques assure correctness, their use for large systems is restricted by memory consumption and computing power. Particularly for an effective solution, the LU decomposition technique will divide the matrix into upper and lower triangular matrices. [4-5]:

$$A = LU \quad (1)$$

Solved utilizing forward and backward substitution, L is a lower triangular matrix and U is an upper triangular matrix.

2.2.2 Iterative Methods

For large systems, iterative methods are more efficient. The methods evaluated include:

Jacobi Method: An iterative technique that updates variables in parallel using an approximation for each element of the system. [6-7]

$$x^{(k+1)} = D^{-1}(b - (L + U)x^{(k)}) \quad (2)$$

Where D is the diagonal matrix, and L and U represent the lower and upper triangular parts of the matrix.

Gauss-Seidel Method: A refinement of the Jacobi method, where updates are done sequentially, often resulting in faster convergence.

Conjugate Gradient (CG) Method: Used for symmetric positive-definite systems, the method minimizes a quadratic function to converge to the solution.

The CG method iteratively solves for x in the equation:

$$Ax = b \quad (3)$$

Where A is a sparse matrix, b is the right-hand side vector, and x is the solution vector.

2.2.3 Preconditioning

To enhance the performance of iterative methods, preconditioning is applied. Preconditioners approximate the inverse of the system's matrix, accelerating convergence. Techniques such as Incomplete LU (ILU) and Jacobi preconditioning were employed to improve the convergence rates of the iterative solvers.

2.2.4 Parallel Processing and High-Performance Computing (HPC)

Due to the scale of the problems, parallel computing was utilized. The implementation of parallelized versions of the Conjugate Gradient and Krylov subspace methods distributed the work across multiple processors, reducing the overall computational time.

2.3 Software and Implementation

The numerical methods were implemented using MATLAB and C++ for efficient memory handling and parallel processing. The PETSc library was used for its robust support of parallel computations in scientific computing. Using multicore CPUs and GPUs for parallel execution, the performance was assessed on a dispersed computing cluster.

3. RESULTS

3.1. Comparison of Direct and Iterative Methods

Using a variety of sparse linear systems, the performance of direct methods (e.g., LU decomposition and Gaussian elimination) as well as iterative techniques (e.g., Conjugate Gradient (CG), Jacobi, and Gauss-Seidel) was assessed. Computational time, memory use, and convergence rate were the main measures of comparison.

LU Decomposition produced precise outcomes but showed excessive memory consumption and sluggish performance, especially for extremely enormous systems. Quadratic growth of computing time with matrix size renders it inappropriate for large-scale projects.

Although Jacobi and Gauss-Seidel methods performed well on somewhat modest systems, they revealed slower convergence for ill conditioned matrices. Though both were considerably more memory efficient than direct methods, the Jacobi method converged slower than the Gauss-Seidel method.

For big, sparse symmetric positive-definite systems, conjugate gradient (CG) showed better results. Particularly when used with preconditioning, the convergence rate was quicker than the iterative approaches. For large systems with 106 variables, the CG algorithm converged in fewer iterations than anticipated, therefore saving considerable time.

3.2. Impact of Preconditioning

The performance of iterative solvers was greatly changed by the use of preconditioning methods. Particularly, the Conjugate Gradient method's convergence was accelerated by the Incomplete LU (ILU) preconditioner, which also lowered the number of iterations by 3050% in poorly conditioned systems. For diagonally dominant systems, the Jacobi preconditioner demonstrated better convergence as well.

The ILU preconditioner proved especially successful in reducing the residual error significantly over fewer iterations in ill conditioned systems.

3.3. Parallelization and High-Performance Computing (HPC) Results

For big sparse linear systems, parallel processing greatly decreased computational time. Compared to single core execution, the CG method implemented in parallel on a distributed computing cluster with multicore CPUs and GPUs demonstrated a speedup factor of 35 times.

With linear scalability noted for systems of up to 107 variables, Parallel Conjugate Gradient (CG) showed very great performance. The synchronization overheads were minimal, and the solution times were reduced proportionally to the number of processors used.

Memory usage was optimized by using Compressed Sparse Row (CSR) format for storing sparse matrices, reducing memory consumption by over 60% for large datasets compared to traditional dense matrix storage formats.

3.4. Convergence Analysis

The convergence rates of all iterative methods were evaluated under different conditions. The results indicated that:

CG method with ILU preconditioning showed the best convergence rate, especially in systems with high condition numbers, achieving near-optimal performance.

The Jacobi method showed linear convergence, which was significantly slower, especially for systems with poor spectral properties.

Gauss-Seidel method exhibited faster convergence than the Jacobi method but still struggled with large, poorly conditioned systems unless preconditioning was applied.

3.5. Accuracy and Stability

The accuracy of the solutions was consistent across all methods, with residual errors within the acceptable range for engineering applications (below 10^{-6} for most systems). The stability of the methods was robust, with iterative methods (particularly CG) maintaining stability for ill-conditioned systems when paired with appropriate preconditioning.

3.6. Summary of Key Results

Direct methods (LU decomposition) are accurate but become inefficient for large sparse systems due to high memory and computational costs.

Iterative methods like CG and Gauss-Seidel are more efficient for large-scale systems, with CG outperforming other methods in terms of both speed and convergence.

The use of preconditioning accelerates convergence, particularly for ill-conditioned systems.

4. DISCUSSION

The direct methods of LU decomposition and Gaussian elimination were highlighted along with several numerical approaches for solving large sparse linear systems in this study. Consequences of preprocessing and parallel processing as well as iterative methods (Jacobi, Gauss Seidel, and Conjugate Gradient). The findings clearly highlight the shortcomings, scalability, and effectiveness of every method used to raise large, thin networks.

4.1. Effectiveness of Direct Methods

Although direct techniques like Gaussian elimination and LU decomposition are well known and guarantee certain results, utilizing these methods to great Particularity as the size of the system grows, sparse linear systems create several issues; LU decomposition demands significant memory and processing capacity; direct methods become computationally costly. For systems with millions of variables, which are impractical due to their great time complexity (typically $O(n^3)$).

Although direct approaches offer a guaranteed solution, the high computational and memory cost makes them unfeasible for big systems. The increased memory use also becomes a constraint as the system size increases, especially in practical applications such computational fluid dynamics and finite element analysis.

4.2. Performance of Iterative Methods

Iterative methods, especially the Conjugate Gradient (CG) method, provided substantial improvements in terms of efficiency for large sparse systems. Unlike direct methods, iterative methods refine an initial guess iteratively, with each iteration bringing the solution closer to the true value. The CG method, in particular, showed excellent performance for symmetric positive-definite systems, converging in fewer iterations than expected, especially when preconditioning was applied. This confirms that CG is well-suited for large systems where direct methods would be inefficient.

The Jacobi and Gauss-Seidel methods also provided reasonable performance for moderately sized systems, but their convergence was slower, especially for ill-conditioned systems. The Gauss-Seidel method converged faster than the Jacobi method, but it still faced limitations when handling large, sparse systems without preconditioning.

In general, iterative methods provided significant memory and computational advantages over direct methods, particularly for systems with sparsity and large sizes. However, their performance is highly dependent on the matrix properties (e.g., symmetry, conditioning) and the choice of preconditioner. [8-9]

4.3. Impact of Preconditioning

Preconditioning is crucial in improving the convergence of iterative methods, particularly for ill-conditioned systems. Preconditioners such as Incomplete LU (ILU) and Jacobi preconditioning significantly reduced the number of iterations required for convergence in iterative methods, particularly the Conjugate Gradient method. This study found that ILU preconditioning accelerated the convergence of CG, lowering the residual error by up to 50% and reducing computational time.

The effect of preconditioning was particularly noticeable in systems with high condition numbers. Without preconditioning in these circumstances, the iterative techniques would have trouble converging or would need many more iterations, which would still be ineffective. Preconditioning reduces these problems by converting the system into a shape simpler to solve, therefore enhancing the general performance of iterative approaches.

4.4. Parallel Computing and Scalability

Parallel computing to address huge sparse systems was one of the main breakthroughs in this research. Particularly in parallelized versions, Conjugate Gradient (CG's) performance was greatly improved. By means of distributed computing clusters and multicore CPUs and GPUs, the computational time was reduced by a factor of 35, mostly for largescale systems with millions of variables.

The parallel implementation showed how well CG and Krylov subspace methods could be scaled to manage huge sparse systems. This results emphasizes how high-performance computing (HPC) could help to solve issues that would be too difficult on single processor computers otherwise. Better memory optimization was also made possible via parallel computing since the sparse matrix storage format (Compressed Sparse Row, CSR) decreased memory use by more than 60%.

Parallel processing gave a major benefit for very big systems since the challenge was divided into smaller sub problems that could be solved at once. The results also suggest that the parallel conjugate gradient approach shows nearly linear scalability for very large systems, therefore it is a possible solution for practical applications needing high computational ability. [10]

4.5. Convergence and Accuracy

The convergence analysis revealed that while direct methods provide exact solutions, iterative methods require fewer computational resources and time, though they may need more iterations to achieve the desired

accuracy. The Conjugate Gradient method, when combined with ILU preconditioning, achieved convergence in fewer iterations, even for large and ill-conditioned systems.

Accuracy was maintained across all methods, with residual errors consistently below 10^{-6} for most of the cases tested, ensuring that the solutions were within acceptable bounds for scientific and engineering applications. Stability was also guaranteed, particularly for iterative methods with appropriate preconditioning. In general, the CG method provided a reliable and accurate solution, even for large and challenging problems.

4.6. Limitations and Future Work

While the results are promising, there are limitations in this study. The scalability of the parallelized CG method was tested on systems up to 107 variables, and future research could explore the performance on even larger systems, potentially up to 108 or more. Additionally, while preconditioning significantly improved performance, the choice of preconditioner must be tailored to the specific properties of the matrix, and future work could explore more advanced preconditioning techniques.

Another area for future research is the integration of multigrid methods and domain decomposition techniques with iterative solvers for even more efficient solutions. As sparse systems are prevalent in areas such as fluid dynamics, structural analysis, and network optimization, further exploration of parallel and distributed algorithms could pave the way for more efficient, real-time solutions for these complex problems.

Parallel processing greatly increases computer speed, hence enabling the solution of big sparse systems in practical uses.

5. CONCLUSION

Highlighted in is the challenges and probably solutions related with resolving large sparse linear systems—which are somewhat prevalent in industries including engineering, scientific computation, and optimization. Particularly on the Conjugate Gradient (CG) approach this study concentrates, We looked at both direct and iterative methods and assessed how much preconditioning and parallel processing accelerated computer speed.

Direct methods, such as LU decomposition, ensure correct results, but their high memory and processing demands make them computer unachievable for huge systems, thus The outcomes are obvious. By contrast, particularly for symmetrical positivedefinite matrices, iterative methods—especially the CG method—provides a more scalable and memoryefficient approach.

Particularly for highly conditioned systems, preconditioning accelerated the convergence rate of iterative methods—particularly the CG method. The Incomplete LU (ILU) preconditioner greatly enhanced the convergence rate of large systems. decreased both computing time and iteration count.

Including parallel processing improved performance even more, allowing the solution of systems with millions of variables. Using multicore CPUs and GPUs greatly lowered solution time, hence enabling the problem to be addressed in realistic, real-world settings.

Finally, for solving massive sparse linear systems, iterative methods combined with good preconditioning and parallel processing offer the best approach. These approaches not only provide scalable solutions but also lower memory consumption and computation time—both of which are absolutely necessary for addressing the difficulties brought on by real-world applications in engineering, scientific simulations, and optimization. Future research might investigate the integration of sophisticated multigrid methods as well as the ongoing development of parallel algorithms to improve sparse system solvers' performance.



ACKNOWLEDGEMENTS

The authors would like to thank the Department of Mathematics, Shahid Madani University, Tabriz, Iran, for providing institutional support to this research.

REFERENCES

- [1] *An Introduction to Sparse Matrices* [Internet]. Springer eBooks, 2023, pp. 1–18. Available: https://doi.org/10.1007/978-3-031-25820-6_1
- [2] T. Lyche, G. Muntingh, and Ø. Ryan, “The Conjugate Gradient Method,” in *Springer*, Cham, 2020, pp. 215–239. Available: https://doi.org/10.1007/978-3-030-59789-4_13
- [3] A. Fevgas, K. Daloukas, P. Tsompanopoulou, and P. Bozanis, “Efficient solution of large sparse linear systems in modern hardware,” in *Proc. Int. Conf. Inf., Intell., Syst. Appl.*, Jul. 2015, pp. 1–6. Available: <https://dblp.uni-trier.de/db/conf/iisa/iisa2015.html#FevgasDTB15>
- [4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 2013.
- [5] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, 2nd ed. Oxford, U.K.: Oxford Univ. Press, 2017.
- [6] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: SIAM, 2003.
- [7] H. A. Van der Vorst, *Iterative Methods for Solving Large Linear Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
- [8] A. Brandt, “Multi-level adaptive techniques for solving elliptic partial differential equations,” *Math. Comput.*, vol. 31, no. 139, pp. 333–390, 1977.
- [9] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid*. San Diego, CA, USA: Academic Press, 2001.
- [10] J. W. H. Liu and S. A. Vavasis, “Parallel solvers for sparse linear systems,” *Comput. Sci. Eng.*, vol. 7, no. 1, pp. 20–30, 2015.

BIOGRAPHIES OF AUTHORS

	<p>Mohammed Farhan Ibrahim is a researcher in the Department of Mathematics at Azerbaijan Shahid Madani University. His research interests lie in the field of numerical analysis, particularly in the development of algorithms for solving eigenvalue problems. Ibrahim's work focuses on improving the efficiency and scalability of iterative methods for large, sparse, and non-symmetric matrices, contributing to advancements in computational mathematics and its applications in science and engineering</p>
	<p>Sahib Abdullah Hasan is a researcher in the Department of Mathematics at Azerbaijan Shahid Madani University. His work focuses on numerical methods for eigenvalue problems, with a particular interest in iterative techniques for large-scale and sparse systems. Dr. Hasan has published research on enhancing computational efficiency in solving matrix eigenvalue problems, contributing to both traditional and modern methods in numerical linear algebra.</p>