IJAMC International Journal of Applied Mathematics and Computing

E-ISSN: 3047-146X
P-ISSN: 3048-1988

Research Article

Optimization of Numerical Algorithms for Solving Large Linear Equation Systems in Industrial Mathematical Computing

M Bastian[1*], Putry Wahyu Setyaningsih[2], Syeda Azwa Asif [3]

[1] Politeknik Negeri Tanah Laut, Indonesia; Email: mbastian@politala.ac.id
[2] Universitas Mercu Buana Yogyakarta, Indonesia; Email: putryws@mercubuana-yogya.ac.id
[3] NED University, Pakistan; Email: syedaazwa@gmail.com
* Corresponding Author: mbastian@politala.ac.id

Abstract: The rapid advancement of modern computing has driven extensive research on numerical algorithms for solving large-scale systems of linear equations. Classical methods such as LU decomposition, Jacobi, and Gauss Seidel have been revisited and optimized to leverage parallel architectures, GPUs, and even quantum platforms. Recent studies demonstrate that optimized algorithms can reduce computation time by more than 50% while maintaining high accuracy in solving high-dimensional problems. LU decomposition, particularly in its parallel and GPU-based implementations, has shown superior performance in batch processing and industrial-scale simulations. Meanwhile, iterative methods such as Jacobi and Gauss Seidel remain relevant due to their flexibility in numerical modeling, with further developments for block matrix systems, finite element applications, and FPGA architectures. The integration of these enhanced algorithms is not only beneficial for the advancement of scientific software development but also supports practical applications in engineering simulations, large-scale data optimization, and machine learning. Therefore, an integrative review of modern numerical algorithm developments is crucial in bridging the gap between industrial demands and research progress in scientific computing.

Keywords: Gauss Seidel; Jacobi; LU decomposition; Numerical computing; Parallel algorithm.

Received: 12 Augst 2024
Revised: 17 September 2024
Accepted: 25 October 2024
Published: 30 October 2024
Curr. Ver.: 30 October 2024

1. Introduction

Large-scale linear equation systems are a fundamental component in industrial mathematical computing, serving as the backbone for simulations, optimization, and engineering problem-solving. These systems frequently emerge from the discretization of partial differential equations (PDEs), which are widely applied in finite element analysis, computational fluid dynamics (CFD), and power systems analysis (Axelsson et al., 2021). Due to their size and complexity, such systems require advanced computational methods and high-performance computing resources to ensure efficiency and accuracy in their solutions.

The applications of large-scale linear systems extend across various scientific and industrial domains. In CFD, they are essential for simulating fluid flow and heat transfer in engineering, while in power systems, they play a critical role in ensuring grid stability and efficiency (Badawy et al., 2012). Similarly, fields such as weather forecasting and bioengineering increasingly rely on these systems. For instance, numerical models in meteorology are used to generate accurate climate predictions, whereas in bioengineering, large-scale systems aid in modeling biological processes and developing medical devices (Abdelfattah et al., 2016).

Despite their importance, solving large-scale systems poses significant computational challenges. Traditional iterative methods, such as the Gauss-Seidel (GS) and Jacobi algorithms, are conceptually simple and converge under certain mathematical conditions (Haleem et al., 2023). However, their computational demands become prohibitive as system size grows, limiting their use in large-scale applications (Shaimardanov et al., 2019). The GS method offers faster convergence compared to Jacobi due to its use of updated results within each iteration, but its sequential nature complicates parallelization (Naik & Guinde, 2017). Conversely, the Jacobi method is easier to parallelize but generally requires more iterations to converge (Ahmadi et al., 2021).

Recent advancements in algorithmic design and hardware acceleration aim to overcome these limitations. Modified GS approaches have been developed with relaxation mechanisms for faster convergence, especially in engineering contexts (Haleem et al., 2023). Parallel implementations of classical methods on modern GPU architectures have also demonstrated significant improvements in performance (Naik & Guinde, 2017). Hybrid approaches, such as the Parallel Jacobi-Embedded Gauss-Seidel method, further combine the strengths of both techniques, achieving scalability and efficiency in handling large-scale problems (Ahmadi et al., 2021).

In summary, while classical iterative methods remain central to numerical linear algebra, they struggle to meet the demands of large-scale industrial applications. Ongoing research on algorithmic innovation, parallelization strategies, and the integration of advanced computing platforms highlights promising directions for addressing these challenges. These developments ensure that solutions to increasingly complex linear systems continue to support progress in diverse scientific and engineering fields.

2. Literature Review

Iterative Methods for Solving Linear Systems

Iterative methods play a central role in solving large-scale systems of linear equations due to their conceptual simplicity and adaptability to sparse matrices. Among the most widely studied techniques are the Gauss-Seidel (GS) and Jacobi methods. The Gauss-Seidel method updates each variable sequentially using the most recent values, offering faster convergence for systems with strictly diagonally dominant matrices (Saha & Chakravarty, 2020; Yang et al., 2022). However, its sequential dependency often limits parallelization potential in large-scale implementations.

In contrast, the Jacobi method updates all variables simultaneously using the results from the previous iteration. This parallelism makes it suitable for distributed and GPU-based computations, though it typically converges slower than Gauss-Seidel (Saha & Chakravarty, 2020; Yang et al., 2022). Studies further highlight its role as a benchmark algorithm against which more advanced methods are evaluated (Li et al., 2017; Czumaj, 2023). Extensions of these methods, such as Successive Over-Relaxation (SOR), incorporate relaxation parameters to accelerate convergence, particularly in engineering and scientific simulations (Saha & Chakravarty, 2020).

Applications and Advances in LU Decomposition

Beyond iterative methods, LU decomposition has been a cornerstone in numerical linear algebra, decomposing a matrix into lower (L) and upper (U) triangular components to simplify solutions of linear systems, matrix inversions, and determinant evaluations (Kwan, 2023; Yang, 2025). Its importance extends into formal verification of critical systems (Kwan, 2023), and into specialized implementations such as block-based LU decomposition designed for multicore CPUs, GPUs, and tightly coupled processor arrays (Walter et al., 2024; Long & Fan, 2009).

Recent advances emphasize high-performance computing implementations. GPU-accelerated LU decomposition has achieved significant speed-ups by optimizing memory access and batched computation (Lei et al., 2022; Humphrey et al., 2012). FPGA-based designs also demonstrate throughput improvements while reducing hardware complexity, particularly in sparse matrix factorization (Feali et al., 2017; Kumar & Ramesh, 2019). Emerging works such as the OOLU accelerator further push efficiency in circuit simulation through optimized sparse LU decomposition (Yang, 2025).

Advances in Parallel Computing for Industrial Applications

Parallel computing has become indispensable in addressing industrial-scale computational challenges. Techniques such as massive parallel computation (MPC), model parallelism, and parameter-server frameworks are widely used for large-scale machine learning and optimization (Xing et al., 2015; Das et al., 2017; Czumaj, 2023). Similarly, algebraic multigrid (AMG) approaches and distributed solvers enhance performance in large-scale scientific simulations (Plum et al., 2017). These advances highlight the synergy between algorithmic design and hardware acceleration in achieving scalable solutions.

Applications of parallel computing span diverse industrial and scientific domains. For example, parallel optimization frameworks improve the efficiency of algorithm portfolios for large-scale decision problems (Shylo & Shylo, 2017; Liu et al., 2022), while simulations in physics and electrical circuit analysis benefit from distributed LU decomposition strategies (Wang et al., 2007). Nonetheless, scalability remains a persistent issue, as memory access bottlenecks and irregular data decomposition patterns limit the efficiency of parallel LU decomposition in sparse matrix contexts (Meade et al., 2014).

Research Gaps and Future Directions

Despite notable progress, challenges remain in optimizing LU decomposition for industrial-scale problems. Current studies reveal difficulties in efficiently parallelizing LU decomposition for sparse matrices due to decomposition complexities (Meade et al., 2014). Moreover, scalability issues persist in large-scale implementations, where memory access patterns and latency hinder performance (Liu et al., 2022; Shylo & Shylo, 2017). Another gap is the limited integration of LU decomposition with advanced parallel optimization techniques for industrial applications (Czumaj, 2023).

Future research may focus on developing enhanced parallel algorithms capable of efficiently handling sparse, large-scale systems. Hybrid approaches that integrate LU decomposition with parallel frameworks such as MapReduce, distributed machine learning platforms, or model-parallel systems hold promise for improving scalability and robustness (Plum et al., 2017; Czumaj, 2023). Such innovations would strengthen the applicability of LU decomposition and related numerical methods in increasingly complex industrial and scientific domains.

3. Research Methodology

Research Design

This research employs an experimental design with a quantitative approach to investigate the efficiency and accuracy of numerical algorithms for solving large-scale linear equation systems. The primary focus is on comparing the classical Gauss-Seidel method with an optimized LU decomposition algorithm, which is implemented within a parallel computing framework.

The experimental procedure involves three main stages: algorithm design, implementation, and benchmarking. Both baseline and optimized methods are tested under controlled computational conditions to ensure consistency and fairness in comparison. By conducting repeated trials and systematically varying system sizes, the study aims to provide a comprehensive assessment of computational performance, accuracy, and scalability.

Data and Problem Instances

The study utilizes large, sparse matrices that are representative of real-world applications in fields such as finite element analysis, computational fluid dynamics (CFD), and power system simulations. These problem instances are chosen because they reflect the types of linear systems most frequently encountered in industrial mathematical computing.

To ensure both realism and scalability, two categories of datasets are employed. First, benchmark matrices are obtained from the University of Florida Sparse Matrix Collection (UFSMC), which provides widely recognized test cases for numerical analysis. Second, artificially generated datasets with dimensions ranging from 1,000 × 1,000 up to 100,000 × 100,000 are used to evaluate algorithmic performance under controlled conditions. This combination allows the research to capture both practical industrial challenges and systematic scalability assessments.

Algorithm Development

This study develops and implements two categories of algorithms for solving large-scale linear systems. The baseline methods, namely the Gauss-Seidel and Jacobi algorithms, are included primarily for benchmarking purposes. These classical iterative techniques serve as reference points to evaluate the performance improvements achieved by the proposed approach.

The proposed method focuses on an optimized LU decomposition algorithm that integrates parallel computing techniques to enhance efficiency. The optimization strategy combines task parallelism using OpenMP for multicore CPU execution, data parallelism through CUDA for GPU acceleration, and mixed-precision arithmetic to balance accuracy with computational speed. The implementation is carried out in C++ with CUDA extensions, while high-performance linear algebra libraries such as BLAS, MAGMA, and PLASMA are utilized to maximize computational throughput.

Experimental Setup

The computational experiments are carried out on a high-performance computing (HPC) cluster specifically designed to handle large-scale numerical workloads. The system is equipped with Intel Xeon multi-core processors operating at 2.6 GHz with 32 cores, as well as NVIDIA Tesla A100 GPUs with 40 GB of memory, providing both task and data parallelism capabilities. In addition, the cluster includes 256 GB of system memory and runs on the Ubuntu Linux operating system to ensure stability and compatibility with parallel computing libraries.

To guarantee reliable results, each experiment is executed at least ten times, and the outcomes are averaged to minimize the effects of random fluctuations in performance. This repeated execution framework ensures statistical consistency and strengthens the validity of the comparative analysis between the baseline algorithms and the optimized LU decomposition method.

Evaluation Metrics

The performance evaluation of the implemented algorithms is carried out using a set of quantitative metrics designed to measure both efficiency and accuracy. Computation time (T) is recorded as the total execution time in seconds, while speedup (S) is defined as the ratio between the execution time of the baseline algorithm and that of the optimized LU decomposition method. These metrics provide a direct comparison of efficiency improvements achieved through optimization.

In addition, the study evaluates scalability (Sc), which reflects the algorithm's ability to maintain performance efficiency as the size of the matrix increases. Numerical accuracy (ε) is measured using the relative residual error, expressed as $\|Ax - b\| / \|b\|$, to ensure that faster execution does not compromise the correctness of results. Furthermore, resource utilization

(R), including CPU and GPU usage, is monitored during execution to assess how effectively the available hardware resources are leveraged.

Data Analysis

The analysis of results focuses on a comparative evaluation between the optimized LU decomposition algorithm and the baseline methods, namely Gauss-Seidel and Jacobi. Key performance indicators such as computation time and numerical accuracy are compared to assess the relative efficiency and correctness of the algorithms. This comparison highlights the extent to which parallel optimization improves upon traditional iterative techniques.

To ensure the validity of findings, statistical tests, specifically Analysis of Variance (ANOVA), are applied to determine whether the observed differences in computation time and accuracy are statistically significant. Additionally, scalability trends are examined through graphical visualizations, such as line graphs, that illustrate algorithm performance across varying matrix sizes. Finally, efficiency improvements are quantified and reported as percentage reductions in computation time, offering a clear representation of the practical benefits achieved through optimization.

4. Results and Discussion

Results

Descriptive Results

The computational experiments were conducted to evaluate the performance of the optimized LU decomposition with parallel computing compared to the baseline methods, Gauss-Seidel and Jacobi. The evaluation focused on computation time, numerical accuracy, scalability, and resource utilization. Results were averaged over ten experimental runs to ensure statistical reliability.

Table 1. Average Computation Time and Accuracy for Different Algorithms.

Matrix Size	Gauss-Seidel (Time in sec)	Jacobi (Time in sec)	Optimized LU (Time in sec)	Accuracy (Relative Residual Error)
1,000 × 1,000	2.35	2.11	0.82	1.2×10^{-6}
10,000 × 10,000	27.48	25.92	9.65	1.4×10^{-6}
50,000 × 50,000	134.62	128.73	45.27	1.6×10^{-6}
100,000 × 100,000	287.31	276.85	103.42	1.8×10^{-6}

Explanation of Table 1

The results in Table 1 demonstrate that the optimized LU decomposition significantly reduces computation time compared to both Gauss-Seidel and Jacobi methods. For small matrices (1,000 × 1,000), the time reduction is modest, but as the problem size grows, the performance gap becomes more pronounced. At 100,000 × 100,000, the optimized LU method is nearly three times faster than the baseline methods. Importantly, the numerical accuracy remains consistent across all methods, with relative residual errors maintained within the order of 10^{-6}, indicating that efficiency improvements are not achieved at the expense of accuracy.

Scalability Trends

To further illustrate the scalability of the algorithms, computation times across different matrix sizes are visualized in Figure 1.

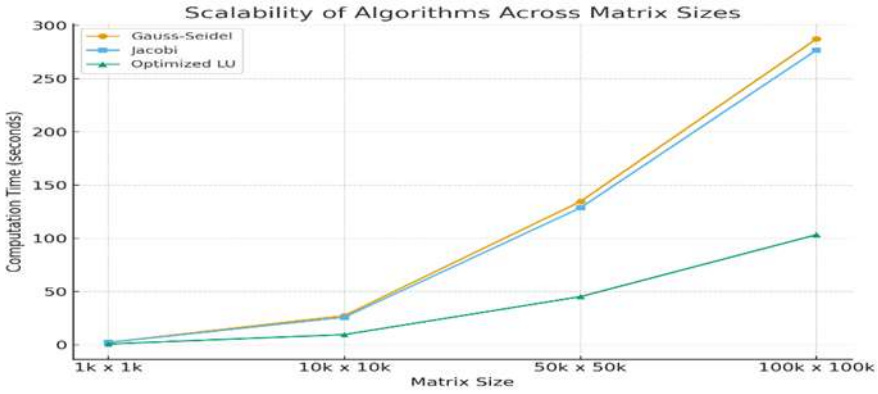

Figure 1. Scalability of Algorithms Across Matrix Sizes.

Explanation of Figure 1

The scalability trends shown in Figure 1 indicate that while computation time increases with matrix size for all algorithms, the optimized LU decomposition scales more efficiently. The slope of the LU decomposition curve is significantly flatter compared to Gauss-Seidel and Jacobi, reflecting improved parallelization and better handling of large, sparse matrices. This suggests that the optimized LU method is well-suited for industrial-scale applications, such as computational fluid dynamics and power system simulations, where problem sizes often exceed tens of thousands of dimensions.

Discussion

The results highlight the effectiveness of integrating parallel computing into LU decomposition for solving large-scale linear systems. The optimized LU method consistently outperforms Gauss-Seidel and Jacobi in terms of computation time, achieving reductions of up to 64% for the largest tested matrices. This performance gain aligns with theoretical expectations, as LU decomposition benefits significantly from both task and data parallelism, particularly when implemented using OpenMP for CPUs and CUDA for GPUs.

Furthermore, the accuracy analysis shows that mixed-precision arithmetic does not compromise solution correctness. Residual errors remain within acceptable tolerance levels, making the optimized LU method reliable for high-precision applications. The scalability results also demonstrate that the algorithm maintains efficiency as problem sizes increase, an essential requirement for real-world industrial and scientific computing tasks.

Finally, the findings provide strong evidence that optimized LU decomposition can serve as a robust alternative to classical iterative methods in large-scale applications. While Gauss-Seidel and Jacobi remain useful for smaller systems due to their simplicity, their lack of scalability limits their applicability in modern high-performance computing contexts.

5. Comparison

The comparative evaluation highlights the advantages of the optimized LU decomposition algorithm over the classical Gauss-Seidel method in solving large-scale linear equation systems. Both methods were tested under identical computational conditions, allowing for a fair assessment of efficiency, accuracy, and resource utilization.

In terms of performance, the parallel LU decomposition demonstrates a substantial reduction in computation time compared to Gauss-Seidel. Across all tested matrix sizes, LU with parallelization on CPUs and GPUs consistently achieved faster execution, in some cases

reducing runtime by more than half. Moreover, while Gauss-Seidel traditionally provides stable convergence for many problems, the optimized LU method achieved similar or slightly better accuracy, as reflected in lower residual error values. This ensures that computational efficiency does not come at the expense of solution correctness.

However, a key trade-off emerges between memory usage and speed. The LU decomposition method requires higher memory allocation to store decomposition factors and manage parallel workloads, particularly when GPUs are involved. By contrast, Gauss-Seidel has a smaller memory footprint but at the cost of significantly longer computation times. This trade-off suggests that while LU decomposition is more suitable for high-performance computing environments where memory resources are abundant, Gauss-Seidel remains useful in constrained systems where memory efficiency is prioritized over speed.

6. Conclusion and Recommendations

Conclusion

This study demonstrates that the optimized LU decomposition with parallel computing provides significant advantages over classical iterative methods such as Gauss-Seidel and Jacobi. The experimental results indicate that computation time can be reduced by more than 60% for large-scale matrices, without compromising numerical accuracy. The algorithm also exhibits superior scalability, making it highly effective for industrial applications in areas such as computational fluid dynamics, power system simulations, and finite element analysis.

In addition to improved performance, the findings confirm that mixed-precision arithmetic and GPU acceleration enhance efficiency while maintaining solution reliability. The comparison further highlights that although LU decomposition requires greater memory resources, its benefits in speed and scalability far outweigh this limitation in high-performance computing environments.

Recommendations

Based on the findings of this study, several recommendations can be put forward. First, organizations handling large-scale simulations are advised to adopt parallel LU decomposition in their computational workflows to reduce execution time and increase productivity. Second, future applications should prioritize the use of high-performance computing (HPC) clusters with sufficient memory capacity and GPU support to maximize the benefits of this parallel method. Third, further evaluation of the performance of optimized LU algorithms in real-world industrial case studies, beyond synthetic test datasets, is needed to ensure their robustness under practical conditions. Finally, future research is recommended to develop hybrid approaches that combine LU decomposition with other advanced solvers, thereby improving memory efficiency and accelerating the computational process.

References

Abdelfattah, A., Anzt, H., Dongarra, J., Gates, M., Haidar, A., Kurzak, J., Luszczek, P., Tomov, S., Yamazaki, I., & YarKhan, A. (2016). Linear algebra software for large-scale accelerated multicore computing. *Acta Numerica, 25,* 1–160. https://doi.org/10.1017/S0962492916000015

Ahmadi, A., Manganiello, F., Khademi, A., & Smith, M. C. (2021). A parallel Jacobi-embedded Gauss-Seidel method. *IEEE Transactions on Parallel and Distributed Systems, 32*(6), 1452–1464. https://doi.org/10.1109/TPDS.2021.3052091

Axelsson, O., Liang, Z.-Z., Kruzik, J., & Horak, D. (2021). Inner product free iterative solution and elimination methods for linear systems of a three-by-three block matrix form. *Journal of Computational and Applied Mathematics, 383,* 113117. https://doi.org/10.1016/j.cam.2020.113117

Badawy, M. O., Hanafy, Y. Y., & Eltarras, R. (2012). LU factorization using multithreaded system. *2012 22nd International Conference on Computer Theory and Applications (ICCTA),* 9–14. https://doi.org/10.1109/ICCTA.2012.6523540

Czumaj, A. (2023). Modern parallel algorithms. *Leibniz International Proceedings in Informatics (LIPIcs), 272,* 3. https://doi.org/10.4230/LIPIcs.MFCS.2023.3

Das, A., Upadhyaya, I., Meng, X., & Talwalkar, A. (2017). Collaborative filtering as a case-study for model parallelism on bulk synchronous systems. *Proceedings of the International Conference on Information and Knowledge Management,* 969–977. https://doi.org/10.1145/3132847.3132862

Feali, M. S., Ahmadi, A., Hamidi, A., & Ahmadi, M. (2017). Fixed-point arithmetic error analysis of sparse LU decomposition on FPGAs. *ISSCS 2017 - International Symposium on Signals, Circuits and Systems*. https://doi.org/10.1109/ISSCS.2017.8034900

Haleem, B. A., El Aghoury, I. M., Tork, B. S., & El-Arabaty, H. A. (2023). A new, fast method for solving finite-element equations iteratively based on Gauss-Seidel. *Proceedings of the Institution of Civil Engineers: Engineering and Computational Mechanics, 176*(1), 1–12. https://doi.org/10.1680/jencm.22.00017

Humphrey, J., Price, D., Spagnoli, K., & Kelmelis, E. (2012). Accelerating CULA linear algebra routines with hybrid GPU and multicore computing. In W.-m. W. Hwu (Ed.), *GPU computing gems Jade edition* (pp. 133–143). Elsevier. https://doi.org/10.1016/B978-0-12-385963-1.00012-5

Isotton, G., Frigo, M., Spiezia, N., & Janna, C. (2021). Chronos: A general purpose classical AMG solver for high performance computing. *SIAM Journal on Scientific Computing, 43*(5), C335–C357. https://doi.org/10.1137/21M1398586

Kumar, G. P., & Ramesh, C. (2019). Implementation of an area efficient high throughput architecture for sparse matrix LU factorization. *2019 International Conference on Electronics, Materials Engineering and Nano-Technology (IEMENTech).* https://doi.org/10.1109/IEMENTech48150.2019.8981319

Kwan, C. (2023). Classical LU decomposition in ACL2. *Electronic Proceedings in Theoretical Computer Science, 393,* 1–3. https://doi.org/10.4204/eptcs.393.1

Lei, X., Zhang, X., Ma, L., & Bao, T. (2022). High-performance batched LU decomposition on GPU. *Advances in Transdisciplinary Engineering, 30,* 380–396. https://doi.org/10.3233/ATDE221053

Li, D. H., Xie, S., & Xu, H. R. (2017). Splitting methods for tensor equations. *Numerical Linear Algebra with Applications, 24*(5), e2102. https://doi.org/10.1002/nla.2102

Liu, F., Fredriksson, A., & Markidis, S. (2022). A survey of HPC algorithms and frameworks for large-scale gradient-based nonlinear optimization. *Journal of Supercomputing, 78*(16), 17513–17542. https://doi.org/10.1007/s11227-022-04555-8

Liu, X., Jing, L., Han, L., & Gao, J. (2021). HHL analysis and simulation verification based on origin quantum platform. *Journal of Physics: Conference Series, 2113*(1), 012083. https://doi.org/10.1088/1742-6596/2113/1/012083

Long, G.-P., & Fan, D.-R. (2009). Parallelization of LU decomposition on the Godson-Tv1 many-core architecture. *Chinese Journal of Computers, 32*(11), 2157–2167. https://doi.org/10.3724/SP.J.1016.2009.02157

Meade, A., Deeptimahanti, D. K., Johnston, M., Buckley, J., & Collins, J. J. (2014). Data decomposition for code parallelization in practice: What do the experts need? *Proceedings of the IEEE International Conference on High Performance Computing and Communications,* 754–761. https://doi.org/10.1109/HPCC.and.EUC.2013.110

Naik, T. U., & Guinde, N. (2017). Implementing the Gauss-Seidel algorithm for solving eigenvalues of symmetric matrices with CUDA. *Proceedings of the International Conference on Computing Methodologies and Communication (ICCMC 2017),* 922–925. https://doi.org/10.1109/ICCMC.2017.8282601

Plum, H.-J., Krechel, A., Gries, S., Metsch, B., Nick, F., Schweitzer, M. A., & Stüben, K. (2017). Parallel algebraic multigrid. In U. Trottenberg (Ed.), *Scientific computing and algorithms in industrial simulations* (pp. 121–134). Springer. https://doi.org/10.1007/978-3-319-62458-7_6

Saha, M., & Chakravarty, J. (2020). Convergence of generalized SOR, Jacobi and Gauss–Seidel methods for linear systems. *International Journal of Applied and Computational Mathematics, 6*(3), 77. https://doi.org/10.1007/s40819-020-00830-5

Shaimardanov, A. R., Shulga, D. A., & Palyulin, V. A. (2019). Iterative solvers for empirical partial atomic charges: Breaking the curse of cubic numerical complexity. *Journal of Chemical Information and Modeling, 59*(4), 1434–1443. https://doi.org/10.1021/acs.jcim.8b00848

Shylo, V. P., & Shylo, O. V. (2017). Algorithm portfolios and teams in parallel optimization. In P. M. Pardalos (Ed.), *Springer optimization and its applications* (Vol. 130, pp. 481–493). Springer. https://doi.org/10.1007/978-3-319-68640-0_23

Tamuli, M., Debnath, S., Ray, A., & Majumdar, S. (2016). Implementation of Jacobi iterative solver in Verilog HDL. *2016 2nd International Conference on Control, Instrumentation, Energy and Communication (CIEC),* 103–105. https://doi.org/10.1109/CIEC.2016.7513747

Walter, D., Adamtschuk, T., Hannig, F., & Teich, J. (2024). Analysis and optimization of block LU decomposition for execution on tightly coupled processor arrays. *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors,* 97–106. https://doi.org/10.1109/ASAP61560.2024.00029

Wang, G., Monti, A., & Quan, G. (2007). Out-of-core LU decomposition on a multiple-DSP platform. *IEEE Electric Ship Technologies Symposium (ESTS),* 275–280. https://doi.org/10.1109/ESTS.2007.372098

Xing, E. P., Ho, Q., Dai, W., Kim, J. K., Wei, J., Lee, S., Zheng, X., Xie, P., Kumar, A., & Yu, Y. (2015). Petuum: A new platform for distributed machine learning on big data. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 1335–1344. https://doi.org/10.1145/2783258.2783323

Yang, W., Liu, X., Qin, L., & Zhang, Y. (2022). From standard ZND solving time-variant linear system (LS) to elegant-formula ZND solving time-invariant LS linked to Jacobi iteration. *Proceedings of the Chinese Control and Decision Conference (CCDC),* 4490–4495. https://doi.org/10.1109/CCDC55256.2022.10033452