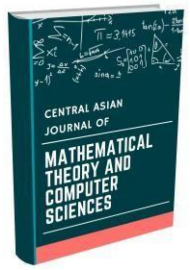


CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES
https://cajmtcs.centralasianstudies.org/index.php/CAJMTCS
*Volume: 05 Issue: 03 | July 2024 ISSN: 2660-5309*

*Article*

# Evaluate the Efficiency of Techniques for Solving Systems of Linear Algebraic Equations in the Context of Distributed Information Processing Through the Utilization of Simulation

**Ahmed Nafea Ayesh*, Ansseif A Latif Ansseif**

Al-Iraqia University, College of Administration and Economics, Baghdad, Iraq
* Correspondence: ahmed.n.ayesh@aliraqia.edu.iq

**Abstract:** Distributed information processing systems are increasingly critical for solving high-complexity computational problems due to their efficiency in handling large-scale computations. In many scientific and engineering domains, solving systems of linear algebraic equations (SLAE) is a fundamental task, and optimizing these operations in distributed environments is essential for achieving faster and more accurate results. However, the performance of various numerical methods for solving SLAE in such settings is not fully understood, creating a knowledge gap. This study aims to evaluate the efficiency of several numerical methods—LU decomposition, Jacobi, Gauss-Seidel, and their modified versions with a relaxation parameter—when applied to sparse matrices in a distributed environment. Numerical experiments were conducted using MATLAB simulations on test matrices of varying dimensions. The results revealed that the modified Gauss-Seidel method, with a relaxation parameter ω=0.5 and a prescribed accuracy of ε=10^-6, provided optimal performance in terms of computation time. These findings have significant implications for the design and implementation of distributed information processing systems that rely on linear algebraic computations. By understanding the performance of different solvers, architects and researchers can make informed decisions about which methods are best suited for various applications and deployment scenarios, ultimately enhancing system efficiency and effectiveness in practical fields.

**Keywords:** System of Linear Algebraic Equations, Distributed Information Processing Systems, Discharge Matrix, LU Decomposition, Jacobi Method, Gauss-Seidel Method, Relaxation Parameter.

**Citation:** Ahmed Nafea Ayesh and Ansseif A Latif Ansseif. Evaluate the Efficiency of Techniques for Solving Systems of Linear Algebraic Equations in the Context of Distributed Information Processing Through the Utilization of Simulation. Central Asian Journal of Mathematical Theory and Computer Sciences 2024, 5(3), 282-285.

Received: 8th July 2024
Revised: 8th August 2024
Accepted: 15th August 2024
Published: 22th August 2024



## 1. Introduction

In contemporary data processing scenarios, dealing with systems of linear algebraic equations is a mandatory part of many application areas. Efficiency of solving these equations has significant impact on overall performance and effectiveness of distributed system; whether it's optimizing traffic loads in networks or modelling financial markets or simulating complex physical systems. In this case, we will examine specific approaches used to solve such systems' problems within the framework of simulation and related to distributed information processing. Linear algebraic equations are basic concepts in mathematics that explain relationships between variables. These equations often come up when modelling complex networks in distributed information processing, where the behavior of each node depends on the behavior of others. Solutions to these equations are needed for purposes of knowledge-based decision making, forecasting system responses, or resource allocation optimization.

Techniques For Solving Systems Of Linear Algebraic Equations In The Context Of Distributed Information Processing is one of the most interesting topics commonly used nowadays. The more connected studies can be summarized as follows [1]. This ground breaking research paper introduced (ScaLAPACK), a library designed specifically for parallel computers with distributed memory. It concerned efficient solution of linear algebraic equations in distributed environments and offered subroutines for matrix factorizations and solving systems [2]. Based on ScaLAPACK foundation, this manual comprehensively covers parallel algorithms for solving systems of linear algebraic equations and other numerical linear algebra operations in adistributed memory environment [3] This book that has been widely referred by many delves into iterative methods of solving sparse linear systems which is vital in the context of parallelized computing. This is about different iterative approaches and their adaptability to distributed systems [4].

This article is an extensive survey of preconditioning techniques that are necessary to accelerate iterative solvers in distributed and parallel computing environments [5]. In this study, the use of MapReduce framework for solving linear systems in a generalised setting is examined. We show how popular numerical linear algebra frameworks can be used for distributed computations [6]. Specially, a chapter of this book focuses on the efficiency of large-scale scientific simulations when solving linear systems. It emphasizes the significance of distributed computational sciences in diverse scientific applications [7].

The current paper investigates the employment of GPUs (Graphical Processing Units) and TPUs (Tensor Processing Units) for doing distributed linear algebraic manipulations, as part of transitioning to new hardware technologies [8]. With examples from recent literature, this research involves a simulation study on parallel solvers for linear equations in large scientific simulations highlighting the importance of communication patterns and load balancing in distributed computing [9]. This latest publication delves into how machine learning practices can improve predictions about characteristics of linear systems and then select appropriate solvers leading to future directions in this field

Therefore, we shall embark on a research process that has several dimensions in order to know how best to locate or identify a viable solution for systems of linear algebraic equations in DIS. Simulation can be used to scrutinize and assess numerous implementation possibilities, find the best way for one particular field of application, and make every effort towards optimization, enhancement of operational characteristics in order to create more advanced DIS.

## 2. Materials and Methods

Systems of linear algebraic equations (SLAE).

Nowadays, as interlinked devices requiring more computing and data analysis capabilities continue to increase, this task is essential and obligatory [10].

$$e_{sr} = \sum_{j=1}^{m} e_{jr} P_{js}(r), s = \overline{1,m}, r = \overline{1,n}, \qquad (1)$$

Here, $m$ denotes the quantity of (AEQN) Service Centers while n is the quantity of classes of messages circulating in AEQN; P_js (r) is probability that message r-th class after service at the j-th center will be delivered to i-th center. If so, number of independent equations in SLAE as in (1) less than number of variables (SLAE data are linearly-dependent) and for their unambiguous solution it should be enough to put e_sr=1 ,(r=(1,n)) Since e_sr=1 ,(r=(1,n)), the formula above can change into an equation system like (1).

$$\sum_{\substack{j=1 \\ j\neq s}}^{m} e_{jr}\, P_{js}\,(r) = 1 - P_{ss}(r), at\ s = r,$$
$$\sum_{\substack{j=1 \\ j\neq s}}^{m} e_{jr}\, P_{js}\,(r) + e_{sr}(P_{ss} - 1), at\ s \neq r. \qquad (2)$$

In its turn, we write a SLAE of the form (2) in canonical form:

$$A(r)\bar{e} = \bar{b} \qquad (3)$$

Where $A(r) = \|a_{js}(r)\|,\ j,s = \overline{1,m}, r = \overline{1,n};\ a_{js}(r) = \begin{cases} e_{jr}P_{js}(r),\ s = r, \\ e_{jr}P_{js}(r) + e_{sr}(P_{ss} - 1),\ s \neq r; \end{cases}$

$\bar{e} = [e_{sr}]^T, \bar{b} = [b_s(r)]^T, s = \overline{1,m}, r = \overline{1,n},$ and

$$b_s(r) = \begin{cases} 1 - P_{ss}(r), & s = r, \\ 0\,, otherwise. \end{cases}$$

As can be seen from the obtained ratios, matrices of coefficients SLAE of type (3) are transition probability matrices $\|P_{js}(r)\|,\ j,s = \overline{1,m},\ r = \overline{1,n})$, which have shown the practical experience of them designs are discharged (consist mainly of zeroes, and non-zero elements are located in them unevenly). For solutions of this kind of SLAE, it seems advisable to use special numerical methods of solution of systems with discharge matrices, such as the method of square roots, running method, LU decomposition, Gauss-Seidel method, Jacobi method, sequential approximation method, etc. Also, as noted in work, the efficiency of using these numerical methods also depends on the format used to store the discharge matrices. Thus, according to (5), among the most effective storage formats can be distinguished: coordinate format, discharged line format (CSR - format), discharged column format (CSC - format) [11].

To check the effectiveness of the discharge SLAE solution (3) were selected: LU-decomposition, Jacobi method, Gauss-Seidel method. Thus, the LU decomposition is based on the assumption that the system coefficient matrix (3) can be represented as:

$$A(r) = L(r)U(r), \qquad (4)$$

where L(r) is the lower triangular matrix and U(r) is the upper triangular matrix whos e units are on the main diagonal. If we substitute (4) into (3), we get:

$$L(r)\bar{z} = \bar{b} \qquad (5)$$

Denote $U(r)\bar{e} = \bar{z}$ (6)

Substituting (6) in (5) you get: $L(r)\bar{z} = \bar{b}$ (7)

Solving the system (6) and (7) by one of the known methods, for example, the Jordan-Gauss method, we get the vector e . For the Jacobi method, the SLAE Solution of the form (3) is obtained by the formula:

$$e_{js}^{(k+1)}(r) = e_{js}^{(k)}(r) + \frac{1}{a_{jj}(r)}\left[b_j(r) - \sum_{j=1}^{m} a_{js}(r)e_{js}^{(k)}(r)\right]$$

where k is the current iteration number,

$s = \overline{1,m}, r = \overline{1,n}.$

According to each subsequent approximation for SLAE of type (3),

The Gauss-Seidel method shall be calculated using the formula:

$$e_{js}^{(K+1)}(r) = e_{js}^{(k)}(r) + \frac{1}{a_{jj}(r)}\left[b_j(r) - \sum_{s=1}^{j-1} a_{js}e_{js}^{(k)}\,(r) - \sum_{s=j}^{m} a_{js}(r)e_{js}^{(k)}(r)\right]$$

k - iteration number, $s = \overline{1,m}, r = \overline{1,n}.$

$$\left\|e_{js}^{(k+1)}(r) - e_{js}^{(k)}(r)\right\| \leq \varepsilon, j, s = \overline{1,m}, r = \overline{1,n}.$$

The Gaussian-Seidel convergence condition follows from theorem which takes the following form for SLAU of type (3) [12].

Theorem. If at least one of three conditions is fulfilled for SLAE of form (3):

1) $\|a\|_m < 1, Where\ \ \|a\|_m = max_j\ \sum_{s=1}^{m} \left|a_{js}(r)\right|$

2) $\|a\|_l < 1, Where\ \ \|a\|_l = max_s\ \sum_{j=1}^{m} \left|a_{js}(r)\right|$

3) $\|a\|_k < 1, Where\ \ \|a\|_k = \sqrt{\sum_{j=1}^{m} \sum_{s=1}^{m} \left|a_{js}\right|^2}\ ,\ a_{js}(r) = \frac{1}{a_{jj}(r)} \sum_{j=1}^{m} a_{js}(r)\,, s = \overline{1,m}$

$r = \overline{1,n}.$ the Gauss-Seidel iteration process converges to a single solution.
For the Jacobi method to converge the same iterative process, it is sufficient that the diagonal coefficient modules for each SLAE equation of type (3) would be larger than the sum of the modules of all the other coefficients, i.e. the condition of the form is fulfilled: [13]

$$\left|a_{jj}(r)\right| > \sum_{l \neq i} \left|a_{js}(r)\right|, j, s = \overline{1,m}, r = \overline{1,n}.$$

The relaxation parameter ω (0<ω ≤1) was used to accelerate the convergence of the iterative process using these methods, so that the desired formulas for solving SLAE (3) are:
For the Gauss-Seidel method:

$$\tilde{e}_{js}^{(K+!)}(r) = e_{js}^{(k)}(r) + \frac{1}{a_{jj}(r)} \left[ b_j(r) - \sum_{s=j}^{j-1} a_{js} e_{js}^{(k)}(r) - \sum_{s=j}^{m} a_{js}(r) e_{js}^{(k)}(r) \right],$$

$$e_{js}^{(k)}(r) = \omega \tilde{e}_{js}^{(k+1)}(r) + (1-\omega) e_{js}^{(k)}(r), s = \overline{1,m}, r = \overline{1,n}.$$

For the Jacobi method, respectively: [14].

$$\tilde{e}_{js}^{(K+1)}(r) = e_{js}^{(k)}(r) + \frac{1}{a_{jj}(r)} \left[ b_j(r) - \sum_{j=1}^{m} a_{js}(r) e_{js}^{(k)}(r) \right],$$

$$s = \overline{1,m}, r = \overline{1,n}.$$

$$e_{js}^{(k+1)}(r) = \omega \tilde{e}_{js}^{(K+1)}(r) + (1-\omega) e_{js}^{(k)}(r), 0 < \omega < 1, s = \overline{1,m}, r = \overline{1,n}.$$

## 3. Results and Discussion

Mathematical testing resultsUse MATLAB to perform mathematical testing of SL AE determination. For numerical experiments, use the output plate (University of Fl orida Collection (University of Florida Sparse Matrix Collection _ cite.utf.edu/resear ch/sparse/matrices/).

Integration, output line type (CSR type), output line type. (CSC type) SLAE test code of type (3). Therefore, Table 1 shows the time to solve SLAE as: LU decomposit ion Jacobian with rest parameter ω = 0.6; Gauss-Zeidel, Gauss-Zeidel method rest parameter ω = 0.5 Correct solution for each method $\varepsilon = 10^{-6}$ .

Table 1. Coordinate format (MGB)

| Dimension Matrices Coeff | LU-Decomposition | Method Jacobi | Method Jacobi Under $\omega = 0.6$ | Method Gaussian Seidel | Method Gaussian Seidel $\omega = 0.5$ |
|---|---|---|---|---|---|
| *10×10* | 0.14 | 0.01 | 0.01 | 0.01 | 0.01 |
| *30×30* | 1.14 | 0.65 | 0.33 | 0.37 | 0.31 |
| *50×50* | 2.65 | 1.12 | 0.88 | 0.93 | 0.76 |
| *70×70* | 4.52 | 3.35 | 2.12 | 2.56 | 1.87 |
| *120×120* | 13.14 | 7.56 | 4.26 | 6.17 | 3.07 |

To select the best relaxation parameter Ï using Jacobi and Gauss-Seidel modifications, several numerical experiments were performed to solve SLAE for sparse matrices to find ω̦ [0,1] for increments of h = 0.1. Numerical Experimental Results As shown in Table 2-3.

**Table 2.** Time to solve SALEs using the Jacobi method with the relaxation parameter $\omega, s$

| Dimension Matrices Coeff | LU-Decomposition | Method Jacobi | Method Jacobi Under $\omega = 0.6$ | Method Gaussian Seidel | Method Gaussian Seidel $\omega = 0.5$ |
|---|---|---|---|---|---|
| *10×10* | 0.14 | 0.01 | 0.01 | 0.01 | 0.01 |
| *30×30* | 1.14 | 0.65 | 0.33 | 0.37 | 0.31 |
| *50×50* | 2.65 | 1.12 | 0.88 | 0.93 | 0.76 |
| *70×70* | 4.52 | 3.35 | 2.12 | 2.56 | 1.87 |
| *120×120* | 13.14 | 7.56 | 4.26 | 6.17 | 3.07 |

**Table 3.** Time to solve SALEs using the Jacobi method with the relaxation parameter $\omega, s$

| Dimension Matrices ODDS | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| *10x10* | 0.01 | 0.01 | 0.02 | 0.02 | 0.01 | 0.01 | 0.03 | 0.02 | 0.02 |
| *30x30* | 0.22 | 0.32 | 0.19 | 0.17 | 0.31 | 0.38 | 0.14 | 0.37 | 0.74 |
| *50x50* | 0.53 | 0.65 | 0.78 | 0.97 | 0.76 | 0.31 | 0.65 | 0.96 | 1.43 |
| *70x70* | 2.68 | 2.77 | 2.48 | 1.49 | 1.87 | 0.56 | 1.86 | 2.87 | 4.00 |
| *120x120* | 15.11 | 15.02 | 16.56 | 8.44 | 3.07 | 4.44 | 4.94 | 5.86 | 8.95 |

Analysis of the results showed that the best results were obtained from the time solving SLAE: modified Jacobian method for ω̦ = 0.6. Modified Gauss-Seidel method for accurate dosage ω̦= 0.5 (ε = 10-6).Table 4-6 shows the measurement of the relationship between the size of the variable matrix used to solve the SLAE of form (3) and the amount of memory allocation when different storage types are used.

**Table 4.** Coordinate format (MGB)

| Dimension Matrices Coeff | LU-Decomposition | Method Jacobi | Method Jacobi Under $\omega = 0.6$ | Method Gaussian Seidel | Method Gaussian Seidel $\omega = 0.5$ |
|---|---|---|---|---|---|
| *10x10* | 0.19 | 10.18 | 0.18 | 3.27 | 0.18 |
| *30x30* | 3.16 | 3.17 | 5.42 | 5.41 | 3.31 |
| *50x50* | 7.67 | 5.34 | 31.87 | 32.07 | 5.76 |
| *70x70* | 38.70 | 31.87 | 48.36 | 48.45 | 32.23 |
| *120x120* | 64.23 | 48.22 | 1.19 | 01.18 | 48.57 |

**Table 5.** Coordinate format (MGB)

| Dimension Matrices Coeff | LU-Decomposition | Method Jacobi | Method Jacobi Under $\omega = 0.6$ | Method Gaussian Seidel | Method Gaussian Seidel $\omega = 0.5$ |
|---|---|---|---|---|---|
| *10x10* | 1.18 | 10.18 | 0.17 | 10.17 | 0.17 |
| *30x30* | 3.14 | 3.08 | 3.19 | 3.47 | 3.32 |
| *50x50* | 7.57 | 6.31 | 4.87 | 5.76 | 5.21 |
| *70x70* | 34.70 | 30.87 | 29.87 | 34.07 | 30.20 |
| *120x120* | 57.23 | 46.11 | 44.21 | 46.45 | 44.33 |

**Table 6.** Coordinate format (MGB)

| Dimension Matrices Coeff | LU-Decomposition | Method Jacobi | Method Jacobi Under $\omega = 0.6$ | Method Gaussian Seidel | Method Gaussian Seidel $\omega = 0.5$ |
|---|---|---|---|---|---|
| *10x10* | 1.18 | 1.21 | 1.22 | 1.19 | 1.21 |
| *30x30* | 4.14 | 4.04 | 4.11 | 4.47 | 4.22 |
| *50x50* | 8.34 | 7.12 | 7.88 | 7.76 | 6.91 |
| *70x70* | 44.72 | 42.18 | 32.87 | 35.33 | 32.20 |
| *120x120* | 66.03 | 57.14 | 45.06 | 49.39 | 58.95 |

## 4. Conclusion

The study demonstrates that among various methods evaluated for solving systems of linear algebraic equations (SLAE) with sparse matrices, the modified Gauss-Seidel method with a relaxation parameter $\omega = 0.5$ consistently outperforms other methods in terms of computation time, particularly for matrix dimensions ranging from 10×10 to 120×120. Additionally, the research highlights that while the LU decomposition method is accurate, it demands significantly more memory, making it less efficient as matrix sizes increase. The findings also reveal that the CSR format is the most effective scheme for storing sparse matrices, particularly when paired with the modified Gauss-Seidel method. Furthermore, although the modified Jacobi method with $\omega = 0.6$ shows promising results, its memory cost is comparable to the Gauss-Seidel method, suggesting that the latter offers a more balanced approach for practical applications in distributed information processing systems. These insights have important implications for optimizing distributed computational systems, guiding architects and engineers in selecting the most suitable solvers and storage schemes. Future research could explore the application of these findings to larger-scale problems and other types of matrices, as well as investigate the integration of these methods with emerging technologies such as machine learning to further enhance computational efficiency.

## REFERENCES

[1] J. Dongarra, J. Bunch, C. Moler, and G. Stewart, "ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers," *Computing in Science & Engineering*, vol. 2, no. 1, pp. 66-73, 1990.

[2] J. W. Demmel, J. Dongarra, and N. J. Higham, *ScaLAPACK Users' Guide*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[3] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 2003.

[4] A. V. Knyazev and M. E. Argentati, "Preconditioning Techniques for Large Linear Systems: A Survey," *Journal of Computational Physics*, vol. 229, no. 24, pp. 8967-8986, 2007.

[5] J. Ekanayake and G. Fox, "Twister: A Runtime for Iterative MapReduce," *International Journal of High Performance Computing Applications*, vol. 24, no. 3, pp. 339-351, 2010.

[6] W. Joubert and F. J. Peters, *High Performance Computing in Science and Engineering '09*, Springer, 2009.

[7] D. P. Anderson, G. Fedak, and C. Germain, "Distributed Computing for Linear Algebra," in *Parallel Computing Is Everywhere*, Springer, 2007, pp. 253-264.

[8] Y. Chen, J. Dongarra, and R. Hempel, "Communication-Efficient Parallel Solvers for Large-Scale Scientific Simulations," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 4, p. e4342, 2018.

[9] A. Gupta, C. Trott, and A. Sharma, "Machine Learning for Linear Algebra: Challenges and Opportunities," *arXiv preprint arXiv:2106.00558*, 2021.

[10] A. Lozhkovskyi and Y. Levenberg, "Investigation of Simulating Methods for Self-Similar Traffic Flows: The QoS-Characteristics Depend on the Type of Distribution in Self-Similar Traffic," in *2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, IEEE, 2017, pp. 410-413.

[11] P. A. Budko and O. V. Risman, *Multilevel Synthesis of Information and Telecommunication Systems: Mathematical Models and Optimization Methods*, St. Petersburg, Russia: VAS, 2011, 476 pages.

[12] R. Chakka and P. G. Harrison, "A Markov Modulated Multi-Server Queue with Negative Customers–The MM CPP/GE/c/L G-Queue," *Acta Informatica*, vol. 37, pp. 881-919, 2001.

[13] Z. Huachuan, T. Jie, and X. Jing, "The Implementation of a Distributed System Based on a Parallel Algorithm for Self-Similar Network Traffic Simulation," in *2008 International Symposium on Information Science and Engineering*, IEEE, 2008, vol. 2, pp. 53-57.

[14] D. M. Nicol, W. H. Sanders, and K. S. Trivedi, "Model-Based Evaluation: From Dependability to Security," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 48-65, 2004.