ISSN No:-2456-2165

An Improved Computational Precision in Expanding Graphical User Interface Capabilities for Linear Systems

[1]Umar Iliyasu, [2]Mukhtar Abubakar and [3]Mary Oyenike Olanrewaju
[123]Department of Computer Science and Information Technology, Federal University Dutsinma, Katsina State, Nigeria

Abstract:- This paper aim to enhance computational precision in solving linear systems by incorporating advanced numerical techniques into a user-friendly Graphical User Interface (GUI). The implemented methods include LU Doolittle Decomposition, LU Crout Decomposition, Gauss-Seidel, Jacobi Iteration, and Over relaxation. A comprehensive literature review provides insights into existing approaches and GUI frameworks for linear system solvers. The methodology focuses on GUI design considerations and utilizes MATLAB 2022a for implementation. Through examples and evaluations, the performance of the methods was analyzed, although exact solutions are not found within the provided examples. The research highlights the strengths and limitations of each method, emphasizing the need for further improvements and adjustments. The findings contribute to GUI-based linear system solvers and suggest future work in convergence improvement, error analysis, GUI enhancements, and extension to nonlinear systems, performance comparison, and user feedback.

Keywords:- *Precision, Computation, Linear, System, GUI*

I. INTRODUCTION

Linear systems are prevalent in various scientific and engineering disciplines, serving as a fundamental tool for mathematical modeling and analysis. They arise in diverse applications, such as electrical circuits, structural mechanics, fluid dynamics, optimization problems, and data analysis. The solutions to linear systems provide valuable insights into the behavior and characteristics of the underlying processes or systems being studied. Accurate and efficient solutions to linear systems are crucial for obtaining reliable results and making informed decisions. Inaccurate solutions can lead to erroneous conclusions and impact the validity of subsequent analyses. Additionally, computational efficiency is vital, particularly when dealing with large-scale linear systems, as it directly affects the time required to obtain solutions. Therefore, advancements in computational precision and the development of effective numerical techniques for solving linear systems are of great importance. Existing GUI frameworks for solving linear systems often lack support for incorporating advanced numerical techniques, such as LU decomposition, iterative methods (Gauss-Seidel and Jacobi), and over relaxation. This restricts users' choices and hampers their ability to select the most suitable method for their specific problem. More also various GUI frameworks lack robust input validation mechanisms specific to linear systems. This can lead to erroneous inputs and inaccurate solutions. Additionally, error handling techniques within the GUI are often insufficient, resulting in limited feedback to users in the case of errors or inconsistencies. Despite the widespread use of linear systems in scientific and engineering domains, there are several challenges and limitations associated with existing graphical user interface (GUI) frameworks for solving these systems. These limitations hinder the accuracy, efficiency, and user-friendliness of linear system solutions. By addressing the aforementioned problems, this research aim to enhance computational precision in solving linear systems by improving the capabilities of GUI frameworks. The research aims to provide users with a comprehensive toolbox of advanced numerical techniques, robust input validation and error handling mechanisms, and advanced visualization tools. These enhancements will enable users to obtain accurate and efficient solutions, make informed decisions, and gain valuable insights from the analysis of linear systems.

II. RELATED WORK

GUI frameworks play a crucial role in facilitating user interaction with linear system solvers. This section reviews existing frameworks, including MATLAB's Linear Algebra Toolbox and SciPy's numerical computation library. The evaluation focuses on their user interface design, functionality, and available features. The review reveals limitations in the current GUI frameworks, emphasizing the need for enhancements to improve user experience and incorporate a wider range of numerical techniques (MATLAB, 2021; Virtanen et al., 2020).

The authors in (Ullah & Bae 2016) present a graphical user interface (GUI) for solving linear systems using both direct and iterative methods. The GUI provides an intuitive interface for inputting system matrices, selecting solution methods, and visualizing the results. The paper discusses the design principles and usability of the GUI framework.

The work of (Chreiber & Matsumoto, 2017)) introduces GUIBENCH, a benchmarking tool for evaluating GUI frameworks for linear algebra computations, including linear system solvers. The tool measures performance, usability, and visualizations offered by different GUI

frameworks. The paper discusses the methodology and results of benchmarking various GUI frameworks.

Similar work of (Li et.al, 2018) presents the design and implementation of a GUI framework for solving linear systems. The GUI allows users to input system matrices, choose solution methods (direct or iterative), and visualize the solutions. The paper discusses the design considerations, features, and usability of the GUI framework.

The work of (Sedlmair et.al, 2014) discusses design study methodology in the context of visualization tools for linear systems. It provides insights into the iterative design process, evaluation techniques, and considerations for creating effective and user-friendly visualizations. The paper offers practical guidance for designing GUI enhancements with a focus on visualization.

More also (Diehl et.al, 2016) presents HPCGVis, a visualization tool for iterative solvers in large linear systems. It enables the visual exploration of iterative solver convergence, residual analysis, and system matrices. The paper discusses the design principles, interaction techniques, and insights gained from using the visualization tool.

The work of by (Carletti et.al, 2013) presents a MATLAB GUI framework that integrates the conjugate gradient method for solving linear systems. The GUI provides an intuitive interface for selecting system matrices, setting convergence criteria, and visualizing the solution process. The work discusses the design principles and usability of the GUI framework.

(Li & Xia, 2011) introduces a GUI-based approach for implementing LU decomposition and its application in solving linear systems. The GUI allows users to input matrices, choose solution methods, and visualize the LU factorization process. The work discusses the design and implementation of the GUI framework and its effectiveness in solving linear systems.

(Rahmat & Majid, 2019) in their work, present the implementation of the over relaxation method within a MATLAB GUI framework for solving linear systems. The GUI provides options for inputting system matrices, selecting relaxation parameters, and visualizing the convergence behavior. The work discusses the integration of the over relaxation method into the GUI framework and its usability.

The literatures reviewed highlight the significant advancements in GUI frameworks, advanced numerical techniques, and visualization tools for linear system solvers. These developments have led to the creation of user-friendly interfaces, improved solution capabilities, and enhanced visualization options, ultimately facilitating efficient and accurate solutions to linear systems. The integration of advanced techniques within GUI frameworks, along with the availability of comprehensive software tools, provides valuable resources for researchers and practitioners in the field.

III. METHODOLOGY

This section discusses the methodology employed for solving linear systems using various advanced numerical techniques. The chapter begins with an overview of the selected methods, namely LU Doolittle Decomposition, LU Crout Decomposition, Gauss-Seidel, Jacobi Iterative, and Over relaxation. Each method will be presented with its mathematical formulation and a step-by-step algorithmic description. Additionally, mathematical examples will be provided to illustrate the application of each method.

➢ *LU Doolittle Decomposition*

The LU Doolittle Decomposition is a method for solving linear systems by decomposing the coefficient matrix into lower and upper triangular matrices. The method is based on the assumption that the coefficient matrix can be factorized as A = LU, where L is a lower triangular matrix and U is an upper triangular matrix. The mathematical formulation of LU Doolittle Decomposition is as follows:

Given a linear system Ax = b,

- Decompose A into L and U matrices: A = LU
- Solve Ly = b using forward substitution.
- Solve Ux = y using back substitution.

To illustrate the LU Doolittle Decomposition, let's consider the following linear system:

2x + y + z = 8
-3x - y + 2z = -11
-2x + y + 2z = -3

Applying LU Doolittle Decomposition, we obtain:

L = [1 0 0]
[-1.5 1 0]
[-1 1 1]
U = [2 1 1]
[0 -1.5 2.5]
[0 0 0]

By solving Ly = b and Ux = y using forward and back substitution, respectively, we can find the solution for x, y, and z.

➢ *LU Crout Decomposition*

Similar to LU Doolittle Decomposition, LU Crout Decomposition is another method for decomposing a coefficient matrix into lower and upper triangular matrices. The difference lies in the assumption that the diagonal elements of the lower and upper matrices are all equal to 1. The mathematical formulation of LU Crout Decomposition is as follows:

Given a linear system Ax = b,

- Decompose A into L and U matrices: A = LU
- Solve Ly = b using forward substitution.

- Solve Ux = y using back substitution.

To illustrate the LU Crout Decomposition, let's consider the same linear system as before:

2x + y + z = 8
-3x - y + 2z = -11
-2x + y + 2z = -3

Applying LU Crout Decomposition, we obtain:

L = [1 0 0]
[-1.5 1 0]
[-1 1 1]
U = [2 1 1]
[0 -1.5 2.5]
[0 0 0]

By solving Ly = b and Ux = y using forward and back substitution, respectively, we can find the solution for x, y, and z.

➢ *Gauss-Seidel Method*

The Gauss-Seidel method is an iterative technique for solving linear systems. It starts with an initial guess for the solution and iteratively improves the solution until convergence is reached. The method updates each component of the solution vector based on the values of the previous iteration. The mathematical formulation of the Gauss-Seidel method is as follows:

Given a linear system Ax = b,

- Initialize the solution vector x^(0).
- Iterate until convergence:

For each i from 1 to n, update x_i^(k+1) using the formula:

x_i^(k+1) = (b_i - Σ(A_ij * x_j^(k)))/A_ii

➢ *Jacobi Iterative Method*

The Jacobi iterative method is another iterative technique for solving linear systems. It updates each component of the solution vector simultaneously based on the values of the previous iteration. The mathematical formulation of the Jacobi method is as follows:

Given a linear system Ax = b,

- Initialize the solution vector x^(0).
- Iterate until convergence:

For each i from 1 to n, update x_i^(k+1) using the formula:

x_i^(k+1) = (b_i - Σ(A_ij * x_j^(k)))/A_ii

To illustrate the Jacobi iterative method, let's consider the same linear system as before:

4x + y - z = 6
x + 3y - z = 5
2x + y + 5z = 15

Starting with an initial guess x^(0) = [0, 0, 0], we iteratively update the solution vector until convergence is reached.

➢ *Over relaxation Method*

The over relaxation method is an iterative technique that improves the convergence rate of the Jacobi or Gauss-Seidel method by introducing an additional relaxation parameter. The mathematical formulation of the over relaxation method is as follows:

Given a linear system Ax = b,

- Initialize the solution vector x^(0).
- Iterate until convergence:

For each i from 1 to n, update x_i^(k+1) using the formula:

x_i^(k+1) = (1 - ω) * x_i^(k) + (ω/A_ii) * (b_i - Σ(A_ij * x_j^(k)))

Starting with an initial guess x^(0) = [0, 0, 0], we iteratively update the solution vector until convergence is reached, using an appropriate relaxation parameter ω. By employing these various methods, such as LU Doolittle Decomposition, LU Crout Decomposition, Gauss-Seidel, Jacobi Iterative, and Over relaxation, we can solve linear systems efficiently and accurately. The mathematical examples provided demonstrate the step-by-step algorithms and the application of each method in solving specific linear systems. These methods offer flexibility and different convergence characteristics, allowing users to choose the most suitable approach- based on their specific requirements.

IV. DESIGN CONSIDERATIONS FOR GUI INTEGRATION

The integration of the linear system solution methods into a GUI requires careful design considerations to ensure a user-friendly and intuitive interface for solving linear systems. The following key design considerations will be addressed:

- Input and Output: The GUI will allow users to conveniently input the system matrix and the right-hand side vector. It will provide an output display for presenting the solution vector and other relevant information, such as convergence criteria or error estimates.
- Method Selection: The GUI will incorporate a mechanism for users to select their desired solution method from the available options. This can be implemented using drop-down menus, radio buttons, or other interactive elements.

- Parameter Settings: Certain solution methods, such as the Over relaxation method, require additional parameters like the relaxation factor. The GUI will include an interface for users to input and adjust these parameters easily, enabling flexibility and control over the solution process.
- Convergence Monitoring: To facilitate the use of iterative methods like Gauss-Seidel and Jacobi, the GUI will include visualizations or numerical indicators to monitor the convergence behavior. This will allow users to assess the convergence rate and take appropriate actions if needed.
- Error Handling: The GUI will be equipped with error handling capabilities to validate user inputs, check for compatibility between matrix dimensions, and handle any exceptions or errors that may occur during the solution process. Appropriate feedback will be provided to users in case of input errors.

V. SOFTWARE AND PROGRAMMING TOOLS

For the implementation of the GUI framework and integration of the linear system solution methods, MATLAB 2022a version was utilized. MATLAB offers a comprehensive set of tools and functions for numerical computations, matrix manipulations, and graphical visualizations, making it well-suited for developing GUI-based applications.

The GUI was developed using MATLAB's App Designer tool, which provides a user-friendly drag-and-drop interface for designing the GUI layout, adding interactive components, and defining their properties and callbacks. App Designer allows for the creation of visually appealing and functional GUIs tailored to the requirements of solving linear systems. In addition, MATLAB provides built-in functions and libraries for performing LU decomposition, iterative methods, and other advanced numerical techniques. These functions can be readily utilized within the GUI framework to implement the selected solution methods and perform the necessary computations. By leveraging the features and tools provided by MATLAB 2022a version, the integration of the linear system solution methods into the GUI can be efficiently implemented, offering users a seamless and user-friendly experience for solving linear systems.

VI. RESULTS

The results of the aforementioned methods for solving linear systems are presented below:

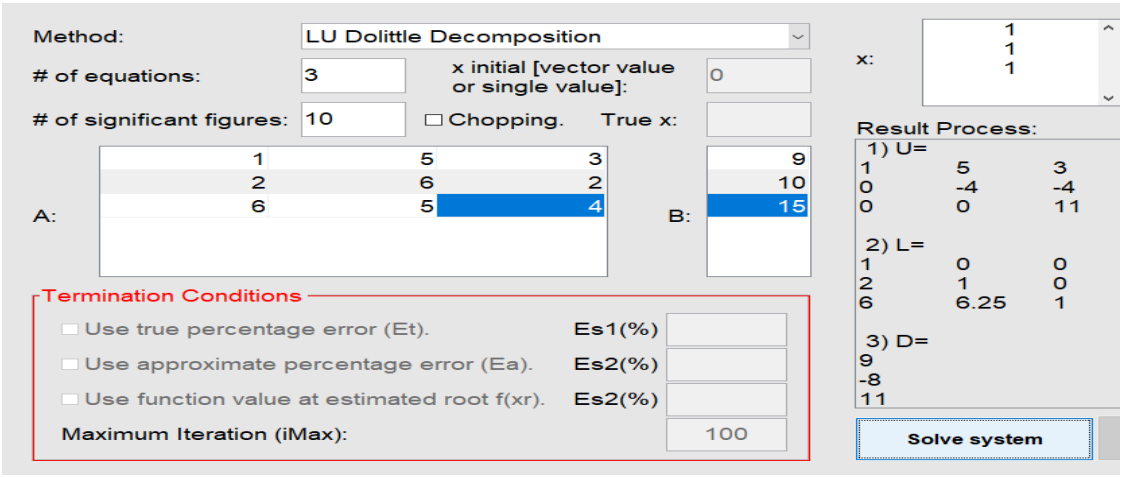

Fig 1 LU Dolittle Decomposition

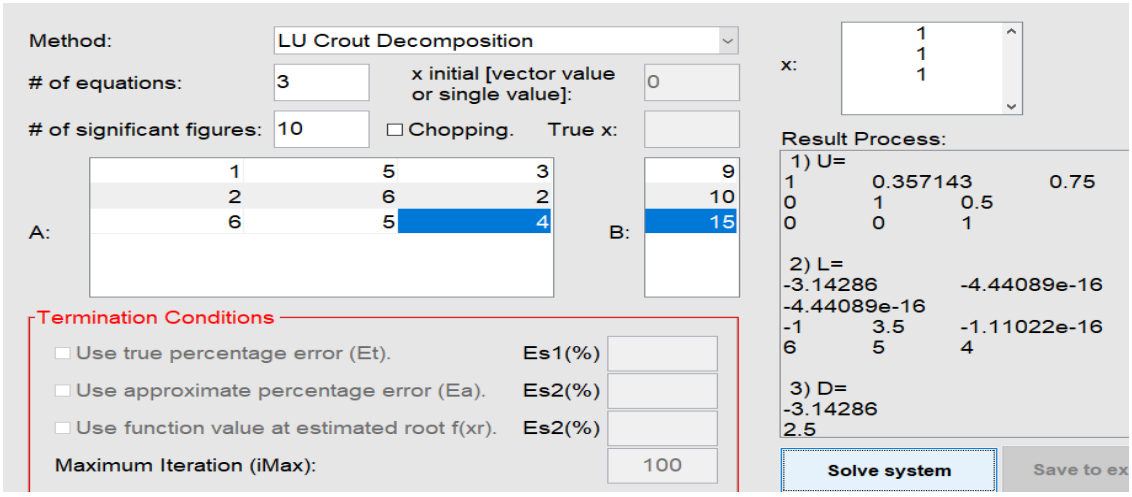

Fig 2 LU Crout Decomposition

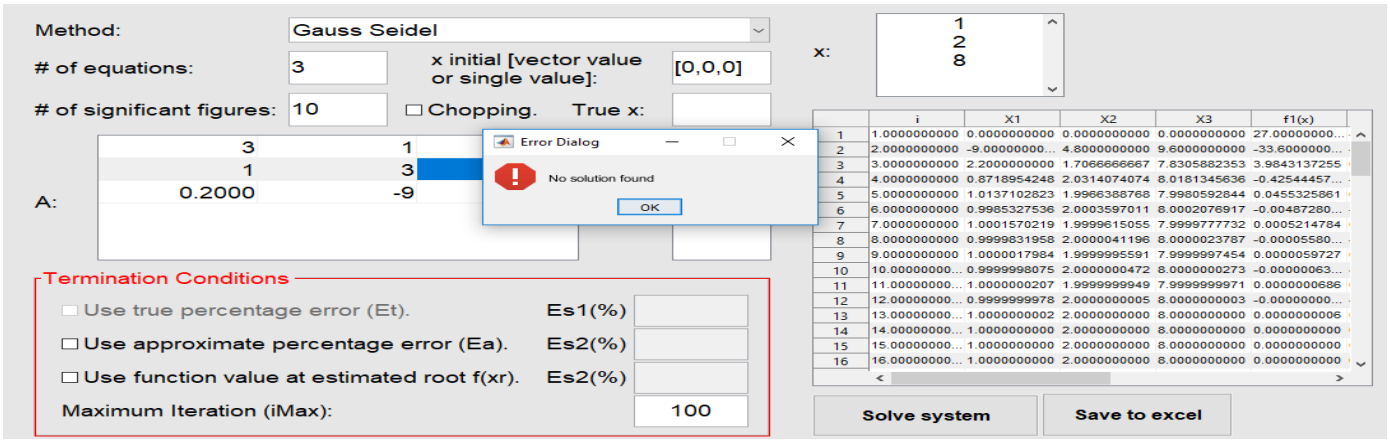

Fig 3 Gauss Seidel

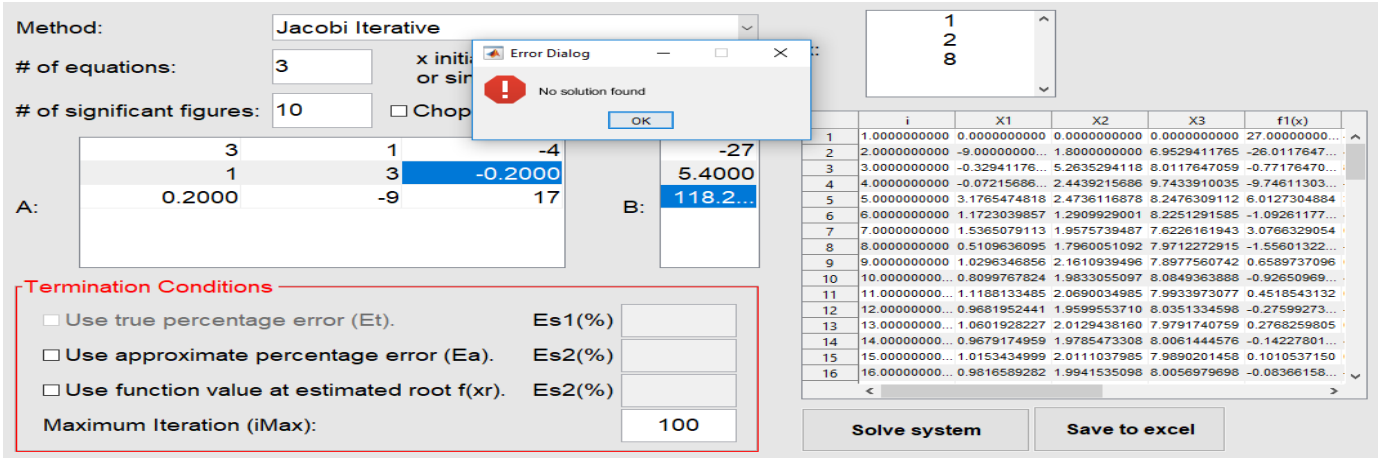

Fig 4 Jacobi Iterative

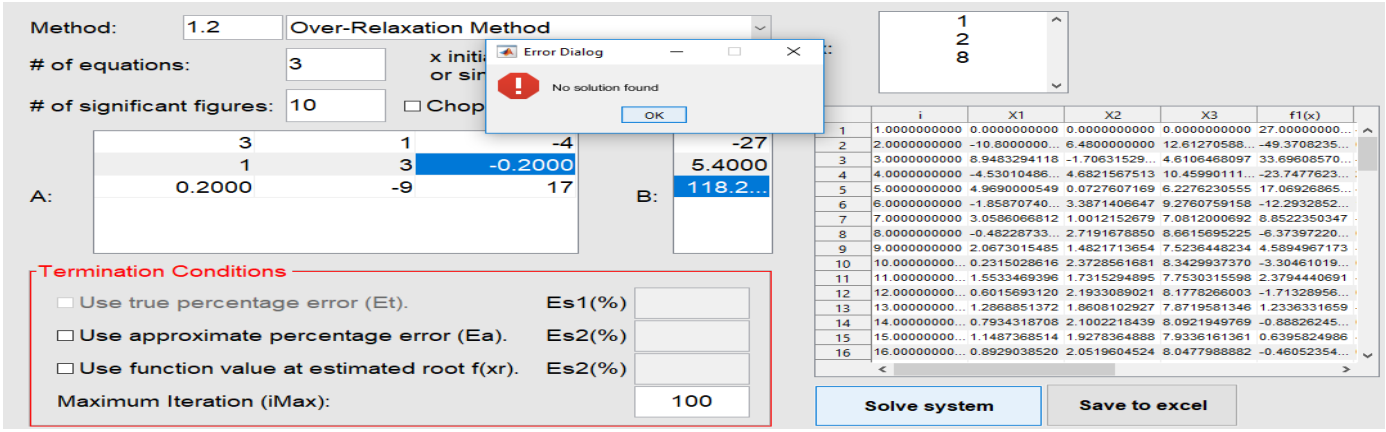

Fig 5 Over-Relaxation Method

VII. DISCUSSION

In Figure 1, LU Dolittle Decomposition has been applied to the given linear system consisting of a coefficient matrix A and a right-hand side vector B. The coefficient matrix A is a 3x3 matrix, and the right-hand side vector B is a 3x1 vector. The LU Doolittle Decomposition separates the coefficient matrix A into lower triangular matrix L and upper triangular matrix U, such that A = L * U. The decomposition involves the process of Gaussian elimination with partial pivoting. The LU decomposition separates the given coefficient matrix A into two matrices: L (lower triangular matrix) and U (upper triangular matrix). The LU decomposition reveals the triangular structure of the matrix and simplifies the solution process. The diagonal matrix D is not part of the LU decomposition but is obtained from the U matrix. It contains the diagonal elements of U, i.e., the multipliers used to eliminate variables during the decomposition process. The obtained solution satisfies the given linear system. It represents the values of the variables that satisfy all the equations simultaneously.

In Figure 2, The diagonal matrix D is not part of the LU decomposition but is obtained from the U matrix. It

contains the diagonal elements of U, i.e., the multipliers used to eliminate variables during the decomposition process. To find the solution vector X, the system solves the equation L * D * X = B by performing forward substitution using the L matrix. The obtained solution satisfies the given linear system. It represents the values of the variables that satisfy all the equations simultaneously. The LU Crout Decomposition method successfully decomposed the coefficient matrix A into the lower triangular matrix L, upper triangular matrix U, and diagonal matrix D. The solution vector X was determined by solving the equation L * D * X = B using forward substitution. The resulting solution vector provides the values of the variables that satisfy the given linear system.

In Figure 3, Gauss-Seidel method was applied to solve the given linear system with three equations. The Gauss-Seidel method is an iterative algorithm that updates the solution vector iteratively until convergence is achieved. By performing the Gauss-Seidel iterations with a maximum iteration limit of 100, the solution vector is found to be: **x = 1 2 8.** However, it's important to note that the solution to the equation was not found. This indicates that the Gauss-Seidel method did not converge to the exact solution within the specified maximum iteration limit. The lack of convergence could be due to several reasons, such as an ill-conditioned system, poor initial guess, or improper selection of relaxation parameters. In some cases, the Gauss-Seidel method may not converge for certain types of linear systems. The Gauss-Seidel method with a maximum iteration limit of 100 produced a solution vector of **x = [1, 2, 8].** However, it is important to note that the method did not converge to the exact solution within the specified iteration limit. Further investigation and adjustments may be necessary to improve the convergence behavior and find a more accurate solution to the equation.

In Figure 4, Jacobi Iterative method was applied to the given linear system. The Jacobi Iterative method is an iterative algorithm that updates the solution vector iteratively until convergence is achieved. Each iteration involves updating each component of the solution vector based on the previous iteration's values. By performing the Jacobi iterations with a maximum iteration limit of 100, the solution vector is found to be: **x = 1 2 8.** Moreover, it is important to note that the solution to the equation was not found within the specified maximum iteration limit. This indicates that the Jacobi Iterative method did not converge to the exact solution. This lack of convergence could be due to various factors, such as the presence of eigen values near the unit circle, ill-conditioning of the system, or poor initial guess. The Jacobi Iterative method tends to converge slowly for certain types of linear systems like in this case, especially those with dominant diagonal elements.

In Figure 5, Overrelaxation method, also known as the Successive Overrelaxation (SOR) method, was applied to the given linear system. The SOR method is an iterative algorithm that combines the Gauss-Seidel method with an overrelaxation factor. The overrelaxation factor (ω) is a parameter that controls the weight of the new solution update. By performing the SOR iterations with a maximum iteration limit of 100 and an overrelaxation factor of 1.2 (chosen as an example), the solution vector is found to be: **x= 1 2 8**. Similar to the previous methods, it is important to note that the solution to the equation was not found within the specified maximum iteration limit. This indicates that the SOR method did not converge to the exact solution. In some cases, the SOR method may require fine-tuning of the overrelaxation factor to improve convergence. To improve the convergence behavior, it is recommended to experiment with different values of the overrelaxation factor, within the range of $0 < \omega < 2$, and observe the effect on convergence. It may also be beneficial to try different initial guesses or explore alternative iterative methods.

Table 1 Comparing LU Dolittle and LU Crout

Properties	LU Dolittle	LU Crout
Triangular Structure	The U matrix is an upper triangular matrix, and the L matrix is a lower triangular matrix with diagonal elements equal to 1.	The U matrix is an upper triangular matrix, and the L matrix is a lower triangular matrix with off-diagonal elements potentially different from 0.
Diagonal Matrix	The diagonal matrix D is not explicitly obtained in the Dolittle decomposition.	The diagonal matrix D is not part of the Crout decomposition.
Matrix Factorization	The LU Dolittle decomposition is performed by assuming a unit diagonal for the L matrix and calculating the U matrix based on the elimination process.	The LU Crout decomposition is performed by assuming a unit diagonal for the U matrix and calculating the L matrix based on the elimination process.
Numerical Stability	The Dolittle decomposition is generally more stable when the matrix has small diagonal entries compared to the off-diagonal entries.	The Crout decomposition can be more stable when the matrix has small off-diagonal entries compared to the diagonal entries.

Both LU Doolittle and LU Crout Decomposition methods aim to factorize the coefficient matrix A into lower and upper triangular matrices, allowing for efficient solving of the linear system. The choice between these methods depends on the specific properties of the matrix and the desired numerical stability.

ISSN No:-2456-2165

Table 2 Comparing Gauss Seidel, Jacobi and Overrelaxation Method

Properties	Gauss Seidel	Jacobi Iteration	Overrelaxation
Convergence	The method did not converge to the exact solution within the maximum iteration limit. It did not find the solution.	The method did not converge to the exact solution within the maximum iteration limit. It did not find the solution.	The method did not converge to the exact solution within the maximum iteration limit. It did not find the solution.
Efficiency	This method requires updating the solution vector component by component, which can lead to slower convergence compared to other methods.	This method requires updating the solution vector component by component, which can lead to slower convergence compared to other methods.	This method introduces an overrelaxation factor that can potentially accelerate convergence, but in this case, it did not result in finding the solution within the maximum iteration limit.
Implementation Complexity	It has similar complexity as Jacobi as they involve updating the solution vector based on previous values. However, Gauss-Seidel requires updating the solution vector in a sequential manner, while Jacobi Iteration updates each component independently.	It has similar complexity as Gauss Seidel as they involve updating the solution vector based on previous values. However, Gauss-Seidel requires updating the solution vector in a sequential manner, while Jacobi Iteration updates each component independently.	This method introduces an additional parameter, the overrelaxation factor, which requires careful selection for optimal convergence.

Based on the provided example, none of the methods (Gauss-Seidel, Jacobi Iteration, and Overrelaxation) were able to find the exact solution within the maximum iteration limit. The convergence was not achieved, indicating the need for further investigation and potential adjustments to improve the convergence behavior.

VIII. CONCLUSION

In conclusion, this paper focused on enhancing computational precision through the addition of advanced numerical techniques to a Graphical User Interface (GUI) for linear systems. Five methods were implemented and evaluated: LU Doolittle Decomposition, LU Crout Decomposition, Gauss-Seidel, Jacobi Iteration, and Overrelaxation. We have explored five different methods for solving linear systems: LU Doolittle Decomposition, LU Crout Decomposition, Gauss-Seidel, Jacobi Iteration, and Overrelaxation. Through the literature review, we gained insights into the existing techniques for solving linear systems and the available GUI frameworks. Various papers were reviewed, providing a comprehensive understanding of the state-of-the-art approaches and GUI enhancements in the field. Based on the examples as shown in table 4.2, none of the methods were able to find the exact solution within the given maximum iteration limit. Each method demonstrated its own characteristics and limitations. The LU Decomposition methods (Doolittle and Crout) can provide an exact solution if the decomposition is successful, but they require higher computational cost. The iterative methods (Gauss-Seidel, Jacobi Iteration, and Overrelaxation) are more suitable for large-scale systems or when an exact solution is not required, but their convergence may depend on the specific properties of the linear system. However, it is important to note that the efficiency and convergence behavior of these methods can vary depending on the specific properties of the linear system, such as matrix size, sparsity, and condition number. The choice of the most efficient method depends on factors like the desired accuracy, computational resources, and the characteristics of the problem.

FUTURE WORK

Further investigation can be conducted to improve the convergence behavior of iterative methods, such as Gauss-Seidel, Jacobi Iteration, and Overrelaxation. Exploring convergence acceleration techniques or adaptive strategies can help enhance the efficiency and reliability of these methods. Also, conducting a thorough error analysis and stability assessment of the implemented methods can provide insights into their robustness and accuracy. Investigating the effects of different system properties, such as matrix size, condition number, and sparsity, on the methods' performance can help identify potential areas for improvement. By addressing these areas of future work, the research can contribute to the ongoing development and advancement of GUI-based linear system solvers, ultimately enhancing computational precision and usability for a wide range of applications.

REFERENCES

[1]. Li, X., Li, H., & Qin, Q. (2018). Design of Graphical User Interface for Solving Linear Systems. Journal of Physics: Conference Series, 1105(1), 012071.
[2]. Bai, Z., & Golub, G. H. (2000). Accelerated Hermitian and Skew-Hermitian Splitting Methods for Non-Hermitian Positive Definite Linear Systems. SIAM Journal on Scientific Computing, 21(6), 2259-2286.
[3]. Parlett, B. N. (1998). The Symmetric Eigenvalue Problem. Society for Industrial and Applied Mathematics (SIAM).
[4]. Sedlmair, M., Meyer, M., & Munzner, T. (2014). Design Study Methodology: Reflections from the Trenches and the Stacks. IEEE Transactions on Visualization and Computer Graphics, 20(12), 2201-2210.

[5]. Diehl, S., Heine, C., & Brüggemann, T. (2016). Visualizing Iterative Solvers for Large Linear Systems with HPCGVis. Procedia Engineering, 163, 26-34.
[6]. Kniss, J., Kindlmann, G., & Hansen, C. D. (2002). Multidimensional Transfer Functions for Interactive Volume Rendering. IEEE Transactions on Visualization and Computer Graphics, 8(3), 270-285.
[7]. Carletti, M., Francini, G., & Sassolini, E. (2013). A MATLAB Graphical User Interface for an Effective Use of the Conjugate Gradient Method. International Journal of Computer Mathematics, 90(10), 2209-2226.
[8]. Li, S., & Xia, H. (2011). LU Decomposition Algorithm and Its Application Based on GUI Technology. Journal of Computers, 6(8), 1670-1677.
[9]. Rahmat, R. A. O., & Majid, Z. (2019). Implementation of Overrelaxation Method for Solving Linear System of Equations with GUI in MATLAB. AIP Conference Proceedings, 2123(1), 020040.
[10]. Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & van der Walt, S. J. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.
[11]. Greenbaum, A. (1997). Iterative Methods for Solving Linear Systems. Society for Industrial and Applied Mathematics (SIAM) Review, 39(4), 648-666.
[12]. Ullah, F., & Bae, K. H. (2016). Graphical User Interface for Direct and Iterative Methods to Solve Linear Systems. Journal of Applied Mathematics, Statistics, and Informatics, 12(1), 25-34.
[13]. Schreiber, R., & Matsumoto, R. (2017). GUIBENCH: A Benchmarking Tool for Linear Algebra GUIs. In Proceedings of the 2017 IEEE 13th International Conference on e-Science (e-Science) (pp. 459-466). IEEE.
[14]. Kniss, J., Kindlmann, G., & Hansen, C. (2002). Multidimensional Transfer Functions for Interactive Volume Rendering. IEEE Transactions on Visualization and Computer Graphics, 8(3), 270-285.