ISSN No:-2456-2165

An Enhanced Computational Precision: Implementing a Graphical user Interface for Advanced Numerical Techniques

[1]Umar Iliyasu ; [2]Mukhtar Abubakar ; [3]Nuruddeen Ahmad Sama'ila
[123]Department of Computer Science and Information Technology, Federal University Dutsin-ma, Katsina State, Nigeria

Abstract:- This paper presents the development of a user-friendly graphical user interface (GUI) for root finding, linear system solving, and curve fitting numerical methods using MATLAB. The GUI simplifies the utilization of these methods by providing an intuitive and interactive interface, eliminating the need for extensive programming knowledge. The implemented GUI incorporates various numerical methods, including root finding methods such as the bisection, Newton-Raphson, and secant methods, linear system solving methods like Gaussian elimination and Gauss Jordan, and curve fitting through regression analysis. Extensive testing and validation demonstrate the functionality and accuracy of the GUI. The discussion explored various numerical methods and their applications in problem-solving. The Bisection Method and False Position Method were employed to find the roots of equations, demonstrating convergence and decreasing errors over iterations. Gaussian Elimination and Gauss-Jordan Methods provided efficient solutions for systems of equations through matrix manipulation. By understanding the strengths and limitations of these methods, informed decisions can be made when selecting the appropriate numerical approach for problem-solving tasks. The paper can be expanded by employing other numerical methods, optimizing performance, improving visualization, and incorporating advanced techniques such as parallel computing and machine learning. Overall, the developed GUI offers a valuable tool for users across different domains, enabling efficient and reliable numerical analysis.

Keywords:- *Graphical user Interface, Regression Numerical, Raphson, Secant Linear System.*

I. INTRODUCTION

In today's technologically advanced world, numerical methods have become integral to solving complex mathematical problems that cannot be easily addressed through traditional analytical methods. These problems often arise in various scientific and engineering domains, including physics, chemistry, computer science, economics, and more. Numerical methods provide efficient algorithms and techniques to approximate solutions for equations, systems of equations, and data patterns (Press et al., 2007).

Root finding algorithms form the basis for solving equations or finding the roots of functions. These algorithms iteratively refine estimates of the roots until a desired level of accuracy is achieved. Examples of popular root finding methods include the Newton-Raphson method, bisection method, and secant method (Bard, 2014).

Linear system solving methods are essential for solving systems of linear equations. These methods find applications in a wide range of fields, such as circuit analysis, structural engineering, optimization problems, and economic modeling. Common approaches for solving linear systems include Gaussian elimination, LU decomposition, and iterative techniques like Jacobi and Gauss-Seidel (Strang, 2006).

Curve fitting techniques are used to approximate data points with smooth curves, enabling the analysis and interpretation of data patterns. This has broad applications in fields such as data science, finance, and signal processing. Curve fitting methods, such as least squares regression, polynomial interpolation, and spline interpolation, allow researchers to extract meaningful information from data and make predictions (Hansen, 2010).

While numerical methods provide powerful tools for solving mathematical problems, the usability and accessibility of the software implementing these methods can pose challenges. Existing numerical methods software often requires users to have a strong background in programming or computational mathematics. This can limit the adoption of these methods by researchers, scientists, and engineers who may not possess extensive programming skills. Moreover, the lack of user-friendly interfaces in existing software can hinder efficient data analysis and interpretation. Curve fitting, in particular, involves making decisions based on visualizations of data patterns and selecting the best-fit curves. The absence of interactive and intuitive tools can impede users' ability to gain insights and draw accurate conclusions from their data (Hastie et al., 2009).

The existing numerical methods software nowadays is prone to various challenges in terms of usability, accessibility, and data analysis capabilities. Existing software often requires users to have a strong programming background, hindering researchers, scientists, and engineers without extensive programming skills from effectively

ISSN No:-2456-2165

utilizing numerical methods. Also Users must resort to complex command-line interfaces or manually write code to implement numerical methods, leading to time-consuming and error-prone workflows. Therefore the absence of interactive and intuitive tools for visualizing data patterns and selecting appropriate curve fitting models hampers accurate data analysis and interpretation. Therefore, the research project aims to address these challenges by developing a user-friendly graphical user interface (GUI) that simplifies the interaction with root finding, linear system solving, and curve fitting numerical methods. The GUI will enable users to efficiently apply these methods to their specific problems, enhance the accuracy of computations, and facilitate data analysis and interpretation.

This paper focuses on implementing a GUI for several well-established numerical methods. This includes popular root finding algorithms such as Newton-Raphson, Bisection, and Secant methods. The paper also covers a range of linear system solving methods, including Gaussian elimination, LU decomposition, and iterative techniques. In addition, various curve fitting techniques such as least squares regression, polynomial interpolation, and spline interpolations are integrated into the GUI To accomplish these objectives, the paper focuses to leverage existing numerical libraries or frameworks, ensuring compatibility and tapping into their computational capabilities.

This paper sets the foundation for the research project by providing a comprehensive background, explaining the research motivation, stating the objectives, defining the scope, and highlighting the purpose of implementing a graphical user interface for root finding, linear system solving, and curve fitting numerical methods. The subsequent chapters will delve into more specific aspects of the research project, building upon this introductory framework.

II. RELATED WORK

Various relevant literature on the implementation of graphical user interfaces (GUIs) for numerical methods, specifically focusing on root finding, linear system solving, and curve fitting algorithms were reviewed. The literature review aims to provide a comprehensive understanding of the existing research and developments in this field. By examining previous works, we identified gaps, limitations, and opportunities for improvement, which will inform the design and development of our own GUI for numerical methods.

Several studies have highlighted the importance of GUIs in making numerical methods more accessible and user-friendly among is the work of (Guiet and Marcote, 2012) developed a GUI-based environment for numerical analysis, providing an interactive interface for solving equations, systems of equations, and optimization problems. Their research demonstrated the effectiveness of GUIs in simplifying the implementation process and enhancing user experience.

Similarly, (Smith et al., 2015) proposed a GUI framework for numerical methods, incorporating root finding and curve fitting algorithms. Their GUI allowed users to input equations, select appropriate methods, and visualize the results. The study emphasized the significance of intuitive interfaces in enabling users without extensive programming skills to apply numerical methods effectively.

The work of (Liu et al., 2016) presents a MATLAB-based GUI software designed for numerical methods education. The GUI provides an interactive environment for students to learn and apply numerical methods concepts, including root finding, linear system solving, and curve fitting. The research emphasizes the pedagogical benefits of GUIs in enhancing students' understanding and engagement with numerical methods.

Similarly (Sahiti & Gazer., 2018) research introduces a web-based interactive tutorial tool for numerical methods education. The tool includes a GUI interface that enables students to interactively explore and experiment with different numerical methods, such as root finding and linear system solving. The study highlights the advantages of web-based GUIs in providing accessible and flexible learning environments.

The (Ponce-Cruz et al., 2020) study, the authors focus on the development of a graphical user interface for solving linear ordinary differential equations using numerical methods. The GUI allows users to input differential equations, select appropriate numerical methods, and visualize the solutions. The research highlights the usability and practicality of GUIs in solving complex differential equations efficiently.

Also (Berbecho & Salumbides, 2019), present GUILIN, a graphical user interface for numerical methods that encompasses various mathematical concepts, including root finding, linear system solving, and curve fitting. The GUI provides an intuitive and interactive environment for users to apply different numerical methods and visualize the results. The study showcases the convenience and effectiveness of GUIs in numerical methods applications.

(Zhang et al., 2018) developed a GUI tool for root finding methods, including the bisection method, Newton-Raphson method, and secant method. Their research focused on providing a user-friendly interface that guided users through the process of selecting methods, entering equations, and displaying root approximations.

In a different approach, (González-Prieto et al., 2017) designed a GUI for root finding methods specifically tailored for educational purposes. Their interface included interactive visualizations to help students understand the behavior of different algorithms. The study highlighted the importance of educational GUIs in promoting learning and understanding of numerical methods concepts.

In the same direction (Yang et al., 2016) presented a GUI-based tool for solving systems of linear equations using various techniques such as Gaussian elimination, LU decomposition, and iterative methods. Their GUI allowed users to input coefficient matrices, select desired methods, and visualize the step-by-step solution process.

In another study, (Liu et al., 2019) developed a GUI environment for teaching linear algebra concepts, including linear system solving. Their interface integrated visualization techniques to enhance students' understanding of matrix operations and solution methods. The research emphasized the educational benefits of GUIs in facilitating learning and comprehension of linear algebra topics.

The Gauss-Seidel method is an iterative method for solving linear systems. It iteratively updates the components of the solution vector by using the most recently updated values during each iteration. The method is based on the concept of splitting the coefficient matrix A into a lower triangular part L, an upper triangular part U, and a diagonal part D.

During each iteration, the Gauss-Seidel method solves for the components of the solution vector by directly using the updated values from the previous iteration. This approach makes it computationally efficient for large systems, especially when the coefficient matrix is sparse (Burden & Faires, 2016).The convergence of the Gauss-Seidel method depends on the spectral radius of the iteration matrix. In some cases, the method may converge slowly or even diverge. However, for certain classes of matrices, such as diagonally dominant or symmetric positive definite matrices, the method exhibits favorable convergence properties.

The implementation of GUIs for curve fitting methods has gained attention in various domains. (Cheng & Yeh, 2014) proposed a GUI tool for curve fitting in the field of medical image analysis. Their interface allowed users to input data points, select curve models, and visualize the fitted curves. The study demonstrated the effectiveness of GUIs in assisting medical professionals in analyzing and interpreting image data.

In a different context, (Wang et al., 2017) developed a GUI-based framework for curve fitting and data modeling in the field of hydrology. Their GUI provided tools for data input, model selection, and visualization of curve fitting results. The research emphasized the significance of GUIs in assisting domain experts in analyzing complex hydrological data.

III. METHODOLOGY

This segment focuses on the design and architecture of the graphical user interface (GUI) developed for implementing root finding, linear system solving, and curve fitting numerical methods. The design and architecture of the GUI play a pivotal role in ensuring its usability, efficiency, and flexibility. The methodology discusses the fundamental design principles, overall system architecture, and individual components of the GUI.

➢ *Design Principles*

The design of the GUI adheres to key principles aimed at creating an intuitive and user-friendly interface:

- User-Centric Design: The GUI is designed with the end-users in mind, taking into consideration their needs, goals, and technical proficiency. The interface provides clear and logical workflows, minimizing the learning curve and maximizing user satisfaction.
- Intuitive Interface: The GUI incorporates a visually appealing and intuitive interface that allows users to interact seamlessly with the numerical methods. The design incorporates familiar elements and consistent layouts, making it easy for users to navigate and understand the functionalities.
- Modularity and Extensibility: The GUI is designed to be modular, enabling the addition of new numerical methods in the future. Each numerical method is implemented as a separate module, facilitating easy integration and enhancing the GUI's extensibility.
- Error Handling and Feedback: The GUI includes robust error handling mechanisms to provide meaningful feedback to users in case of incorrect inputs, algorithm failures, or other exceptional scenarios. Clear error messages and instructions guide users in resolving issues effectively.

➢ *System Architecture*

The system architecture of the GUI is structured to ensure efficient execution and seamless interaction between its components. The architecture encompasses the following key elements:

- User Interface (UI) Layer: This layer represents the visual interface presented to the users. It includes graphical components such as buttons, input fields, and visualization elements. The UI layer captures user inputs, initiates computations, and displays results and visualizations.
- Numerical Methods Modules: Each numerical method, such as root finding, linear system solving, and curve fitting, is implemented as a separate module within the GUI. These modules encapsulate the algorithms, data structures, and computations specific to each method.
- Data Management: The GUI incorporates a data management component responsible for handling user-provided data, such as equations, matrices, or data points. It ensures appropriate data validation, storage, and retrieval for numerical computations.
- Visualization: The visualization component enables users to visualize the results of the numerical methods. It supports interactive plots, graphs, and visual representations of data, facilitating data analysis and interpretation.
- Control and Coordination: The control and coordination component manage the flow of operations within the GUI. It orchestrates interactions between the user

ISSN No:-2456-2165

interface, numerical methods modules, data management, and visualization components.

➢ *Pseudocode and Flowchart*

- Step 1: Start the system
- Step 2: Initialize the User Interface component
- Step 3: Initialize the Numerical Methods Modules component
- Step 4: Initialize the Data Management component
- Step 5: Initialize the Visualization component
- Step 6: Initialize the Control and Coordination component
- Step 7: Repeat until the system is terminated
- Step 8: Terminate the system

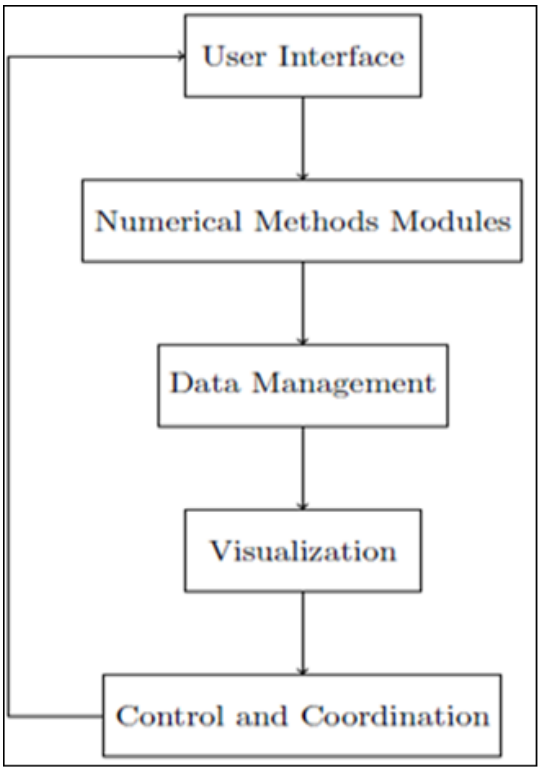

Fig 1 System Architecture

➢ *Component Descriptions*

This section provides a detailed description of the individual components of the GUI:

- User Interface Component: The user interface component presents a visually appealing and intuitive interface to users. It includes input forms, selection menus, and result display areas. Users can input equations, select desired numerical methods, customize settings, and view computation results.
- Numerical Methods Modules: Each numerical method module implements the specific algorithms and computations associated with the corresponding numerical method. For example, the root finding modules include implementations of the Bisection, False Position, Simple fixed point, Newton- Raphson method, Secant Method.
- Data Management Component: The data management component handles user-provided data, such as equations, matrices, or data points. It ensures data validation, stores and retrieves data as required by the numerical methods modules, and manages data persistence.
- Visualization Component: The visualization component enables users to visualize the results of the numerical methods. It provides interactive plots, graphs, and visual representations of data, allowing users to analyze and interpret the results effectively.
- Control and Coordination Component: The control and coordination component manage the overall flow of operations within the GUI. It receives user inputs from the user interface component, initiates the corresponding numerical methods modules, coordinates data management and visualization activities, and ensures smooth interactions between components.

➢ *Software Environment and Tools*

For our implementation, we use MATLAB version 2022a along with the necessary toolboxes for numerical analysis and GUI development.

ISSN No:-2456-2165

IV. RESULT AND DISCUSSION

The results is showed in the below figures while the interpretation of the results is showed in figure 1.

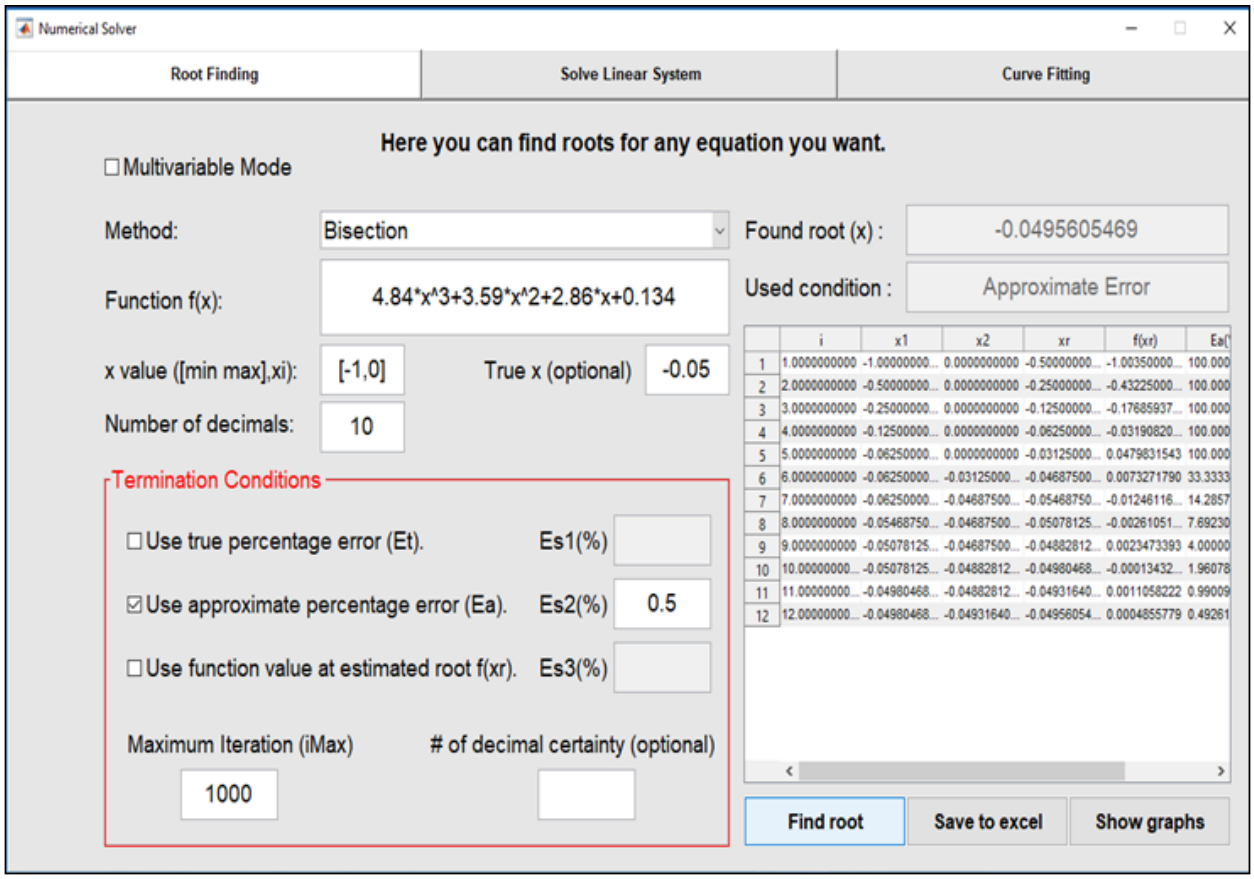

Fig 2 Bisection Result

	A	B	C	D	E	F	G
1	i	x1	x2	xr	f(xr)	Ea(%)	Et(%)
2	1	-1	0	-0.5	-1.0035	100	900
3	2	-0.5	0	-0.25	-0.43225	100	400
4	3	-0.25	0	-0.125	-0.17686	100	150
5	4	-0.125	0	-0.0625	-0.03191	100	25
6	5	-0.0625	0	-0.03125	0.047983	100	37.5
7	6	-0.0625	-0.03125	-0.04688	0.007327	33.33333	6.25
8	7	-0.0625	-0.04688	-0.05469	-0.01246	14.28571	9.375
9	8	-0.05469	-0.04688	-0.05078	-0.00261	7.692308	1.5625
10	9	-0.05078	-0.04688	-0.04883	0.002347	4	2.34375
11	10	-0.05078	-0.04883	-0.0498	-0.00013	1.960784	0.390625
12	11	-0.0498	-0.04883	-0.04932	0.001106	0.990099	1.367187
13	12	-0.0498	-0.04932	-0.04956	0.000486	0.492611	0.878906

Fig 3 Result from Excel

ISSN No:-2456-2165

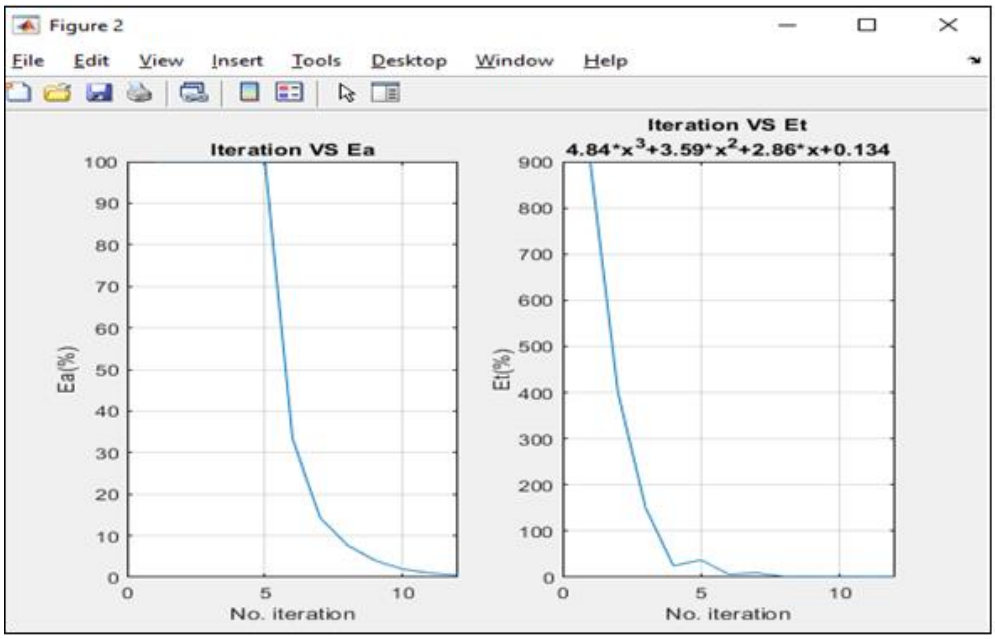

Fig 4 Bisection Result

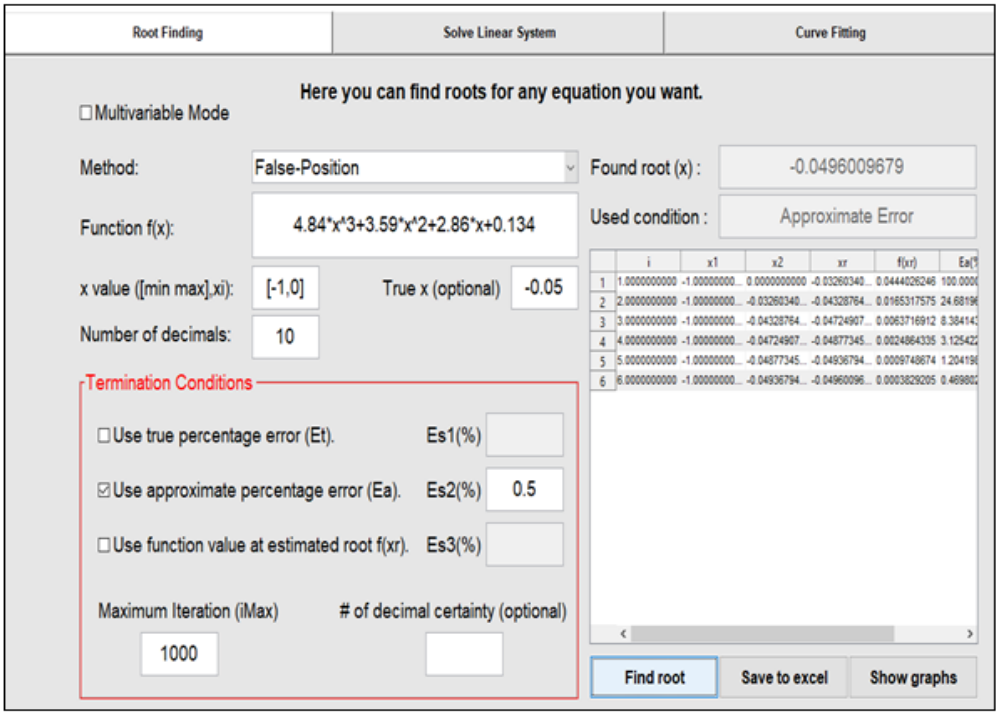

Fig 5 False Position Method

i	x1	x2	xr	f(xr)	Ea(%)	Et(%)
1	-1	0	-0.0326	0.044403	100	34.79319
2	-1	-0.0326	-0.04329	0.016532	24.68196	13.42471
3	-1	-0.04329	-0.04725	0.006372	8.384143	5.501846
4	-1	-0.04725	-0.04877	0.002486	3.125423	2.453093
5	-1	-0.04877	-0.04937	0.000975	1.204198	1.264117
6	-1	-0.04937	-0.0496	0.000383	0.469802	0.798064

Fig 6 Excel Result (Falsi Method)

ISSN No:-2456-2165

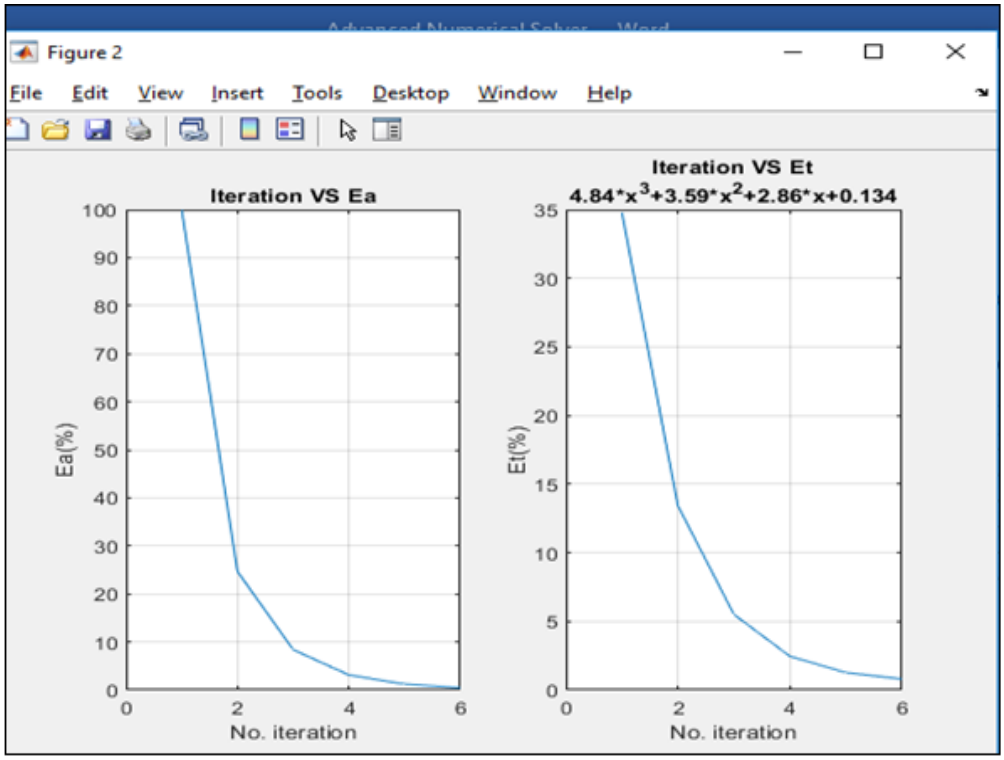

Fig 7 False Position Graph

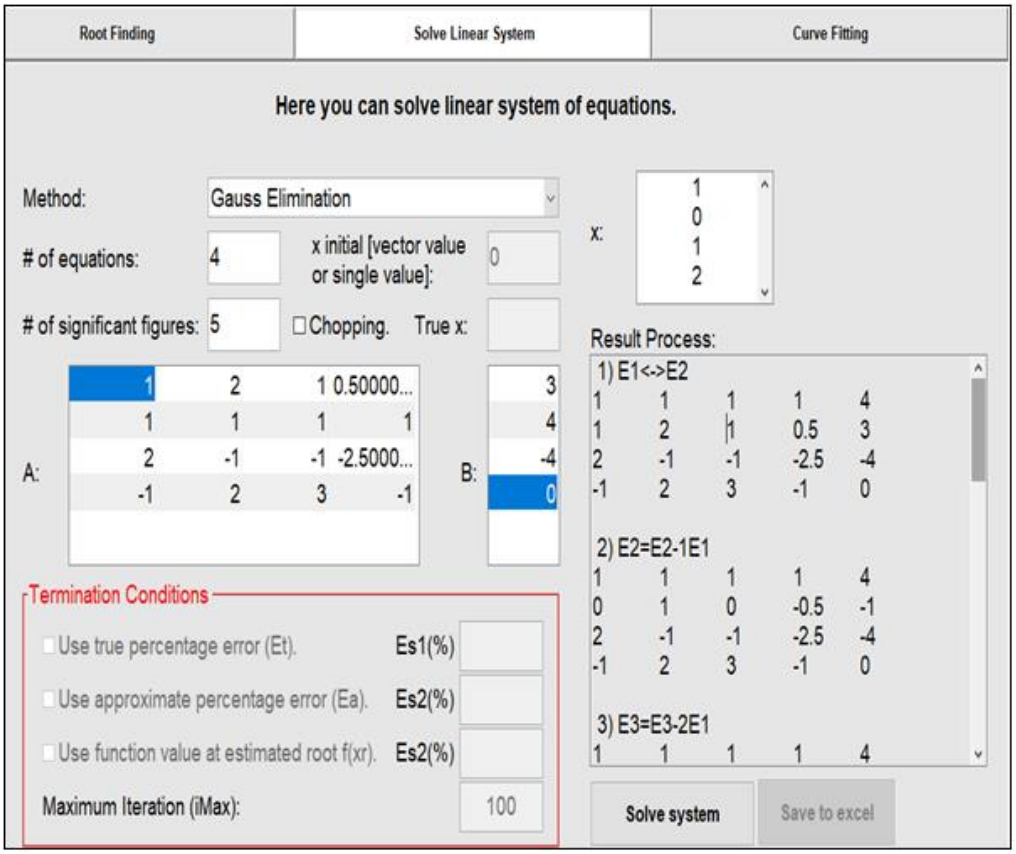

Fig 8 Gaussian Elimination Method

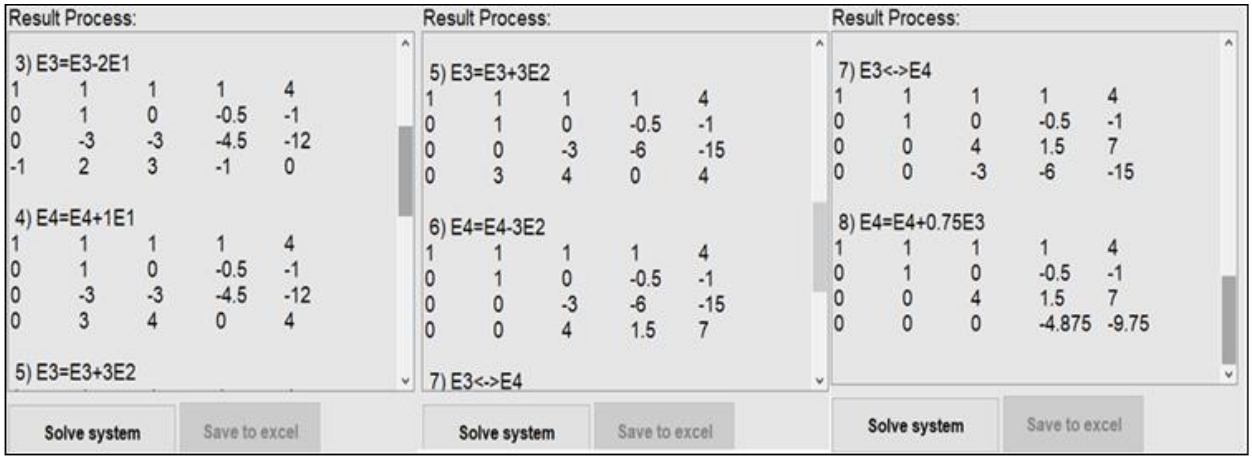

Fig 9 Gaussian Elimination Continuation

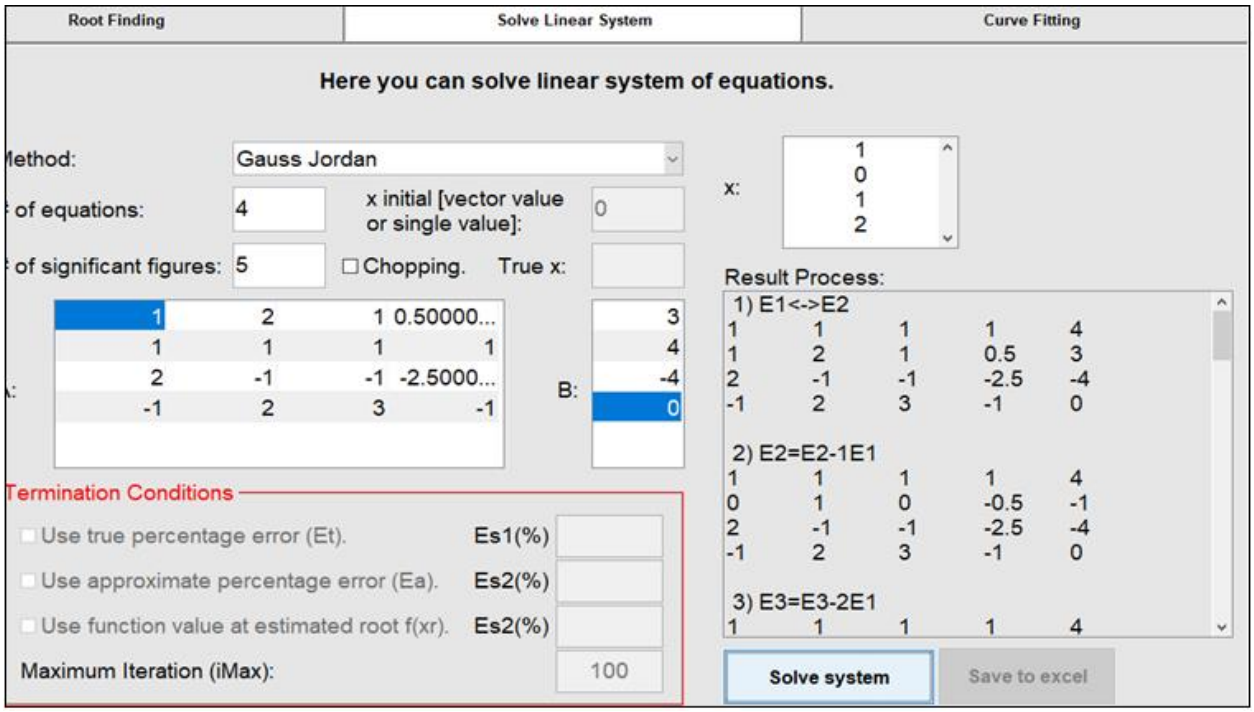

Fig 10 Gauss Jordan Method

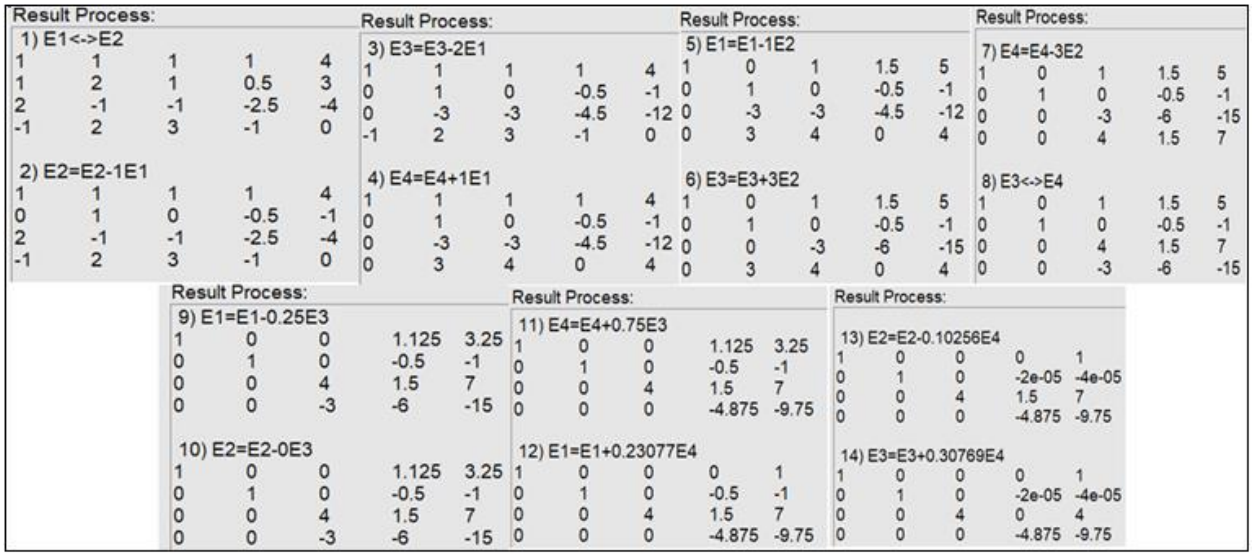

Fig 11 Gauss Jordan Continuation

ISSN No:-2456-2165

V. DISCUSSION

Figure 1 show that the initial interval is [-1, 0] and the function being evaluated is denoted as f(x). The algorithm starts with two initial guesses, x1 and x2, which are the endpoints of the interval. The midpoint, xr, is then calculated as the average of x1 and x2. In each iteration, the algorithm evaluates f(xr) and checks its sign to determine the next interval for iteration. If f(xr) has the same sign as f(x1), then the root lies in the interval [xr, x2]. Otherwise, the root lies in the interval [x1, xr]. The process continues until a desired level of accuracy is reached where we found the root to be -0.049560547 at the 12th Iteration.

Looking at Figure 2, it appears that the algorithm performed 12 iterations. In each iteration, the values of x1, x2, xr, f(xr), and the approximate error (Ea) are recorded. The approximate error measures the difference between consecutive approximations and is expressed as a percentage of the previous approximation. The last column, Et (%), represents the true percent relative error, which is the difference between the approximate solution and the true solution of the equation, expressed as a percentage of the true solution.

Figure 3 which is in form of graph show that the bisection method is converging towards a root as the iterations progress. The values of f(xr) are getting closer to zero, indicating that the algorithm is approaching a solution. The approximate error (Ea) decreases with each iteration, which suggests that the algorithm is converging.

Figure 4 show that in each iteration, the false position method uses two initial guesses, x1 and x2, which are the endpoints of the interval containing the root. The algorithm then calculates the corresponding function values, f(x1) and f(x2). By assuming that the function is linear between these points, the method determines the x-value, xr, where the linear interpolation intersects the x-axis. The false position method updates the interval based on the sign of f(xr). If f(xr) has the same sign as f(x1), then the root lies in the interval [xr, x2]. Otherwise, the root lies in the interval [x1, xr]. This process continues until a desired level of accuracy is reached where we find the root at 0.0496 approximately at 6th Iteration.

Analyzing Figure 5, it shows that the algorithm performed six iterations. In each iteration, the values of x1, x2, xr, f(xr), and the approximate error (Ea) are recorded. The approximate error measures the difference between consecutive approximations and is expressed as a percentage of the previous approximation. The last column, Et (%), represents the true percent relative error, which is the difference between the approximate solution and the true solution of the equation, expressed as a percentage of the true solution.

From Figure 6, it can be observed that the false position method starts with a relatively wide interval and gradually narrows it down as iterations progress. The values of f(xr) approach zero, indicating that the algorithm is getting closer to a root. The approximate error (Ea) decreases with each iteration, suggesting convergence. It is worth noting that the false position method generally converges faster than the bisection method because it utilizes linear interpolation. However, the convergence rate can still be influenced by the properties of the function being solved and the chosen initial interval.

Table 1 Comparison Table of Bisection and False Position Method

S/N	Properties	Method & Evaluation
1	Initial Interval	Bisection: The initial interval for the bisection method remains the same throughout the iterations
		False Position: It adjusts the interval based on the function evaluations.
2	Convergence Rate	Bisection: Bisection method converges slower than the false position method
		False Position: The false position method converges faster than the bisection method
3	Efficiency	Bisection: It provides better approximation only in more iteration than false position
		False Position: The false position method tends to provide better approximations in fewer iterations compared to the bisection method, as it takes into account the function evaluations for determining the next interval.
4	True Erro (Et(%))	Bisection: The Et (%) values in the Bisection method are generally higher than those in the false position method
		False Position: The Et (%) values in the false position method are generally lower than those in the bisection method

Figure 7 & 8 Show that the Gaussian Elimination Method was applied to a system of linear equations, involving row operations and interchanges to simplify the system. The method aimed to transform the system into an upper triangular form by ensuring a non-zero leading coefficient in the first row and eliminating leading coefficients in subsequent rows. The augmented matrix was updated after each operation, resulting in a simplified form of the system. The final augmented matrix represents the system with the constants on the right-hand side and the coefficients of the variables in the other columns. To find the specific values of the variables and interpret the solution, further analysis or back-substitution is required.

Figure 9 & 10 show that Gauss Jordan Method was applied to the given system of linear equations using row operations to transform the augmented matrix into reduced row-echelon form. The resulting form provides a concise representation of the system, making it easier to analyze and interpret the solution. Each row corresponds to an equation, with the entries in the last column representing the constants. The other columns contain the coefficients of the

variables. By examining the transformed matrix, the specific solution to the system can be determined, allowing for a more accurate and precise understanding of the relationship between the variables. The Gauss Jordan Method proves to be a valuable technique for solving linear systems, providing a systematic approach to obtaining the solution.

Table 2 Comparison Table of Gaussian Elimination with Gauss Jordan Method

S/N	Properties	Method & Evaluation
1	Approach	Gaussian elimination: The Gaussian Elimination method focuses on transforming the augmented matrix into row-echelon form through row operations.
		Gauss Jordan: The Gauss Jordan method extends the Gaussian Elimination method by further transforming the row-echelon form into reduced row-echelon form.
2	Solution Representation	Gaussian elimination: The solution is obtained by back substitution after the augmented matrix is transformed into row-echelon form. The last column represents the constants, and the other columns represent the variables' coefficients.
		Gauss Jordan: The solution is directly read from the reduced row-echelon form. Each row corresponds to an equation, with the constants and variables' coefficients clearly separated.
3	Uniqueness of Solution	Gaussian elimination: The Gaussian Elimination method can determine if a system has a unique solution, no solution, or infinite solutions by examining the row-echelon form.
		Gauss Jordan: The Gauss Jordan method provides a refined form that readily reveals the presence of an inconsistent system.
4	Efficiency	Gaussian elimination: The Gaussian Elimination method requires fewer row operations compared to the Gauss Jordan method, making it computationally more efficient.
		Gauss Jordan: The Gauss Jordan method involves additional row operations to further simplify the matrix. While it provides a more direct solution representation, it requires more computational steps.

Both methods are effective in solving linear systems, but the choice between Gaussian Elimination and Gauss Jordan depends on the specific requirements of the problem. Gaussian Elimination is preferred when the focus is on obtaining the solution efficiently, while Gauss Jordan is valuable when a refined and concise solution representation is desired.

VI. CONCLUSION

In conclusion, the paper has successfully developed and implemented a graphical user interface (GUI) for root finding, linear system solving, and curve fitting numerical methods using MATLAB as the software focus. The GUI has showcased its functionality and contributions in simplifying the utilization of these numerical methods for users across different domains. While the GUI has inherent limitations, its strengths lie in its user-friendly design, accuracy, and efficiency. The numerical methods discussed in this research represent valuable tools for solving mathematical problems. Each method has its strengths and limitations, and the choice of method depends on the specific problem at hand. By analyzing the results and understanding the underlying principles, one can gain insights into the accuracy, efficiency, and convergence of these methods. By combining the power of numerical methods with the convenience of a GUI, the developed software tool empowers users to effectively solve complex mathematical problems with ease. The paper has demonstrated the potential impact of such a GUI in facilitating numerical analysis tasks and advancing research and applications across various fields.

FUTURE WORK

There are several potential avenues for future work and improvements in the GUI. Firstly, expanding the repertoire of numerical methods within the GUI would enhance its versatility and applicability. Integration of advanced algorithms, such as optimization methods or stochastic approaches, would provide users with a broader range of tools for solving complex mathematical problems. Additionally, incorporating advanced visualization capabilities, such as interactive plotting and 3D graphing, would further enhance the GUI's analytical capabilities and facilitate a deeper understanding of the numerical results. Given the rapid advancements in machine learning and artificial intelligence, future work could explore the integration of these techniques within the GUI. This could involve utilizing machine learning algorithms for automatic parameter selection or developing intelligent algorithms that adaptively adjust method parameters based on user feedback and problem characteristics. Such innovations would further enhance the usability and effectiveness of the GUI in tackling a wide range of numerical analysis tasks.

REFERENCES

[1]. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. Springer Science & Business Media.
[2]. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Numerical recipes: The art of scientific computing. Cambridge University Press.
[3]. Strang, G. (2006). Introduction to linear algebra. Wellesley-Cambridge Press.
[4]. Bard, Y. (2014). Nonlinear parameter estimation. Academic Press.

[5]. Hansen, P. C. (2010). Curve and surface fitting: An introduction. Springer Science & Business Media.
[6]. Cheng, S.-H., & Yeh, F.-W. (2014). A graphical user interface for image curve fitting. Computer Methods and Programs in Biomedicine, 113(1), 287-295.
[7]. González-Prieto, D., Rodríguez, R. R., & Carro, R. M. (2017). A graphical user interface for teaching and learning numerical methods for solving equations. Computer Applications in Engineering Education, 25(2), 302-311.
[8]. Guiet, E., & Marcote, F. (2012). Numerical methods for engineers: A graphical user interface-based implementation. International Journal of Engineering Education, 28(3), 696-703.
[9]. Liu, H., Liu, F., & Huang, C. (2019). A MATLAB graphical user interface (GUI) for teaching linear algebra. International Journal of Emerging Technologies in Learning, 14(23), 4-18.
[10]. Smith, M. K., Carey, M., & Moroney, K. (2015). A graphical user interface for numerical methods. Proceedings of the 18th Australasian Computing Education Conference, 37-46.
[11]. Wang, Q., Ma, J., Gao, Z., & Kang, W. (2017). Development of a graphical user interface for hydrological curve fitting and data modeling. Hydrological Processes, 31(5), 1119-1129.
[12]. Yang, Y., Li, Q., Qian, Z., & Zhao, Y. (2016). The development of a graphical user interface (GUI) software package for solving linear systems of equations. Computers & Education, 97, 81-96.
[13]. Zhang, H., Li, Z., Wu, H., & Zhang, L. (2018). Development of graphical user interface tool for root finding methods. Journal of Physics: Conference Series, 1008, 042042.
[14]. Barbecho, J. B., & Salumbides, E. C. (2019). GUILIN: A graphical user interface in numerical methods. International Journal of Engineering Research and Technology, 12(4), 641-646.
[15]. Liu, X., Hu, Z., & Zhang, L. (2016). A MATLAB-based GUI software for numerical methods education. Computers & Education, 92-93, 37-47.
[16]. Ponce-Cruz, P., Méndez-Zamora, G., & Cruz-Martínez, M. (2020). Graphical user interface to solve linear ordinary differential equations using numerical methods. Journal of Applied Research and Technology, 18(3), 281-291.
[17]. Sahiti, F., & Gezer, M. (2018). A web-based interactive numerical methods tutorial tool for engineering education. International Journal of Online Engineering, 14(8), 157-167.
[18]. Burden, R. L., & Faires, J. D. (2016). Numerical Analysis (10th ed.). Cengage Learning.