Using SciLab for Solving a set of Linear Equations.

Pamula Rao Mathe

Department of Mathematics, Govt. Degree College, Repalle, Andhra Pradesh, India.

Received on 17th October 2024; Revised on 6th November 2024; Accepted on 19th November 2024

Abstract

Software tools designed for numerical analysis are widely applied in both academia and scientific inquiry. These tools range from proprietary applications to freely available open-source alternatives. Their adoption is largely driven by the challenges associated with complex mathematical operations, especially those involving linear computations. Additionally, the growing presence of multiple variables in both linear and nonlinear expressions necessitates efficient computational solutions. This study set out to explore key pedagogical strategies, instructional techniques, and creative approaches for delivering linear equation concepts at the tertiary education level. By integrating this framework into the curriculum, educators can foster improved comprehension of topics like solving systems of linear equations—an essential skill for students pursuing engineering disciplines. In response to these needs, the investigation presented Scilab as a practical, low-cost programming environment suitable for numerical computation. The proposed use of Scilab includes its application in modeling mathematical problems. These structured techniques could then be repurposed as instructional resources for math-focused courses.

INTRODUCTION

Conveying concepts in numerical analysis and simulation-based modeling often presents instructional hurdles—particularly when addressing procedures like variable elimination in systems of linear equations. A frequent obstacle among learners is their struggle with traditional algebraic foundations, especially when required to convert symbolic mathematical expressions into computer code. In addition, handling complex mathematical operations by hand poses significant constraints, highlighting the necessity for novel and varied approaches to hands-on learning in computational mathematics.

To put it differently, numerical computation serves as a practical means for resolving advanced mathematical problems. These include both linear and nonlinear equations, operations in calculus such as differentiation and integration, differential equations, numerical series, and approximation errors. A variety of algorithms—such as matrix-based techniques like LU factorization, Gaussian and Gauss-Jordan methods, matrix inverses, and iterative solvers like Jacobi and Gauss-Seidel—are widely used to address linear equation systems.

In recent years, the intricacy of such linear systems has increased, particularly with the addition of multiple variables in both linear and nonlinear functions. As a result, manual calculations are insufficient, making computational tools essential. While several software platforms are available for solving these problems—such as Excel, MATLAB, and LINDO—most of them are commercial products, requiring paid licenses for authorized use. [1,2].

Despite the widespread use of proprietary programs, a variety of no-cost and open-source platforms are available for performing numerical computations with reliable precision. In this context, the current discussion highlights Scilab as a strong candidate for handling advanced linear mathematical challenges. What sets Scilab apart is its accessibility—it is a freely available and open development tool, unrestricted by licensing fees or commercial limitations, making it a budget-friendly alternative [3–5].

Past studies have delved into Scilab's applications in artificial intelligence tasks. For example, research in [6] assessed Scilab's effectiveness for time series prediction using neural network architectures. The study tested models like ANN-GD with enhanced backpropagation, ANN-GA utilizing genetic algorithms, and ANN-DE integrated with differential evolution strategies. Datasets included Hyndman's historical time series (January 1961–October 1975) and monthly employment records from Wisconsin (January 1962–December 1975). The experiments confirmed Scilab's potential in predictive analytics, highlighting both its simplicity and economic advantage.

In another investigation, Catarino P. and Vasco P. [3] applied Scilab within the educational domain to support instruction in linear algebra. Their findings suggested that Scilab's command structure made it easier for students to express complex mathematical equations, thereby improving overall classroom interaction and student involvement.

Moreover, a study conducted by Yusop N.M.M., Hasan M.K., and Rahmat M [8] employed version 5.4.0 of Scilab to address systems of ordinary differential equations (ODEs). They utilized a modified form of Euler's method—known as Harmonic Euler—to solve these equations. Their results further support Scilab's utility as a cost-effective programming environment for applying numerical techniques.

METHODOLOGY

Introducing to Scilab Programming

Scilab is a high-level programming environment designed to handle a wide range of problems in scientific and numerical computing. Initially launched in 1990 by INRIA (Institut National de Recherche en Informatique et en Automatique) in France, it was created to support advanced mathematical computations. As a freely accessible tool, Scilab shares functional traits with commercial platforms like MATLAB and Minitab but comes without the associated costs.

Compatible across multiple platforms—including Windows, macOS, and Linux—Scilab can be freely obtained from its official site at www.scilab.org/products/scilab/download. The software is released under the CeCILL open-source license, allowing unrestricted use without licensing expenses. One of its notable features includes seamless compatibility with LabVIEW, a graphical system-design software.

For editing and writing code, Scilab incorporates a built-in text editor known as Scinotes. Users can launch Scinotes directly through the console or by navigating to the Application > Editor option in the interface. This built-in tool enhances the scripting experience, offering a convenient space for coding and debugging within the Scilab environment. [4].

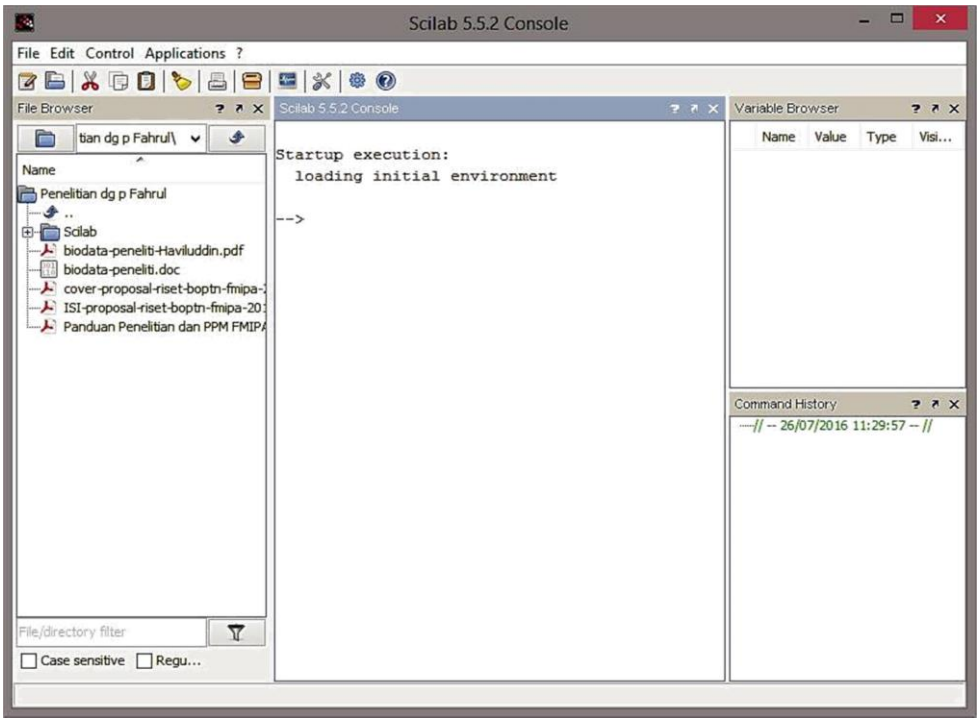

Fig.. The Scilab Console

Computational Techniques in Applied Mathematics

Computational approaches in mathematics are employed to construct matrices for resolving systems of equations, compute ordinary differential equations (ODEs), and carry out numerical integration tasks. These methods are grounded in mathematical modeling frameworks [10,11]. This article outlines four fundamental techniques used for numerical analysis: the Gaussian Elimination approach, Gauss-Jordan reduction, matrix inversion, and LU (Lower-Upper) factorization.

Gaussian Elimination Approach

Credited to the mathematician Carl Gauss, this approach simplifies systems of equations through systematic transformation of a matrix into a triangular (or echelon) form using elementary row operations. The method also involves appending a column for constants and applying row manipulations to reach a solution [3,12].

Gauss-Jordan Reduction Technique

An extension of the standard Gaussian method, Gauss-Jordan reduction carries the row operations further to derive a matrix in reduced row echelon form. This method provides more refined results compared to its predecessor by continuing the transformation until each leading variable is isolated [3,12].

Matrix Inversion Technique

Another common strategy is to utilize the inverse of a coefficient matrix to find solutions to linear systems. While not as computationally efficient as Gaussian methods, it becomes useful when repeatedly

solving equations with different constant terms but the same coefficient structure [12]. As stated by MathWorks in MATLAB documentation, "The inverse function outputs a matrix of the same type unless the original is non-invertible, in which case it returns a failure result. Symbolic output is provided if the input isn't a matrix" [13].

LU Factorization Method

LU decomposition offers an alternative version of solving matrix equations derived from Gaussian Elimination but skips some steps during transformation. These skipped processes are later reinstated during decomposition into two triangular matrices: one lower and one upper [12]. As described in MATLAB's official documentation, "The LU function decomposes a matrix A into the product of two triangular matrices—typically a permutation of a lower triangular matrix and an upper triangular matrix. This factorization is often referred to as LU or LR and works with rectangular matrices as well" [13].

RESULTS AND DISCUSSIONS

In this investigation, Scilab programming was applied to reinforce foundational problem-solving concepts commonly encountered in engineering education. Various computational techniques—namely Gaussian Elimination, Gauss-Jordan, Matrix Inversion, and LU Factorization—were selected as representative examples for implementation. A prior study conducted by Fahrul Agus and Haviluddin assessed the performance of these four numerical strategies using Scilab. Their findings confirmed that consistent, functional procedures could be developed within Scilab to execute these algorithms effectively [14]. The source codes they developed for these methods are accessible in their published work [14], and additional scripts covering other techniques are available upon request from the current study's authors.

The algorithms originally introduced by Agus and Haviluddin were refined in this research and applied to a broader range of numerical techniques commonly covered in Numerical Analysis courses. These enhancements demonstrated that Scilab is highly adaptable and can be leveraged as instructional content for subjects like Mathematics and Statistics.

Conclusions

Scilab proves to be a highly capable tool for demonstrating computational methods in numerical analysis. Its open-access nature and intuitive structure make it especially suited for illustrating concepts such as solving linear systems. Given its accessibility and ease of use, Scilab is an excellent educational resource for both instructors and learners, and it continues to receive widespread support from the academic community for teaching computational mathematics.

REFERENCES

1. Salleh Z., Yusop M.Y.M., Ismail S.B. Basic of Numerical Computational Using Scilab Programming. in the 2nd International Conference on Mathematical Applications in Engineering (ICMAE2012). pp. 1-8 (2012).
2. Baudin M., Couvert V. Optimization In Scilab. The Scilab Consortium - Digiteo / INRIA (2010).
3. Catarino P., Vasco P. Scilab in Linear Algebra. Applied Mathematical Sciences, vol. 8 2014, No. 28 pp. 1391-1399 (2014).
4. Baudin M., Steer S. Optimization with Scilab. present and future. pp. 1-8 (2009).

5. Wang Y., Liu L., Cao J., Feng K., Fu J., Li C. Random Forests Toolbox with Scilab and its Application. in The 7th International Conference on Computer Science & Education (ICCSE 2012) July 14-17 2012., Melbourne, Australia, pp. 1082-1085 (2012).

6. Panigrahi S., Karali Y., Behera H.S. Time Series Forecasting using Evolutionary Neural Network. Int J Comput Appl T (0975 – 8887), vol. 75– No.10, August 2013, pp. 13-17, (2013).
7. Singh A. Implementation of Back-Propagation Neural Network using Scilab and its Convergence Speed Improvement. Int. Journal of Electrical & Electronics Engg., vol. 2, Spl. Issue 1 (2015), pp. 192-194, (2015).
8. Yusop N.M.M., Hasan M.K., Rahmat M. Comparison New Algorithm Modified Euler in Ordinary Differential Equation Using Scilab Programming. Lecture Notes on Software Engineering. 3(3): 199-202, (2015).
9. Desmaële D., Renaud L., Tingry S. A wireless sensor powered by a flexible stack of membraneless enzymatic biofuel cells. Sensors and Actuators B, 220: 583–589 (2016).
10. Rao S. Applied Numerical Methods for Engineers and Scientist 3rd Edition (2002).
11. Salleh Z. Fundamental of Numerical Methods for Scientists and Engineers, 1st Edition (2011).
12. Urroz G.E. Matrix Operations with SCILAB. (2001).
13. MathWorks. LU Matrix Factorization. (2016).
14. Fahrul Agus and Haviluddin AIP Conference Proceedings 1813, 040005 (2017)