A Novel Approach to Solve Numerical Methods Using Matlab Programming

Diya Mehra, Priyanka Sond, Reena Agnihotri
GNA University, Phagwara, Punjab
Corresponding author: Reena Agnihotri, Email: reena.agnihotri@gnauniversity.edu.in

Numerical techniques is a branch of mathematics that explores problem-solving methods that account for approximation errors. This paper defines the algorithms for some of the most popular iterative procedures for solving polynomial problems, including the Bisection Method, Newton- Raphson Method, Secant Method, and Gauss-Jacobi Method and many more. The purpose of this research is to use MATLAB software to solve numerical method problems. The primary goal of this study is to describe and resolve various numerical methodologies using a single platform. We also provide a brief overview of different methodologies and their underlying algorithms, which may make them easier to comprehend.

Keywords: Bisection, MATLAB, Gauss-Seidel, Secant, Simpson's

2023. In Vikram Dhiman & Pooja Dhand (eds.), *Emerging Trends in Engineering and Management*, 109–119. Computing & Intelligent Systems, SCRS, India.
https://doi.org/10.56155/978-81-955020-3-5-12

1. Introduction

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment [1]. MATLAB is a very powerful software package that has many built-in tools for solving problems and for graphical illustrations [2]. MATLAB stands for MATrix LABoratory. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics. The MATLAB software package offers a setting in which we can study programming and investigate the architecture of numerical algorithms [4].

2. Bisection Method

For the solution of polynomial equations, use the bisection method. The interval containing the equation's root is divided and separated by it. The guiding principle for this method is the intermediate theorem for continuous functions The proper answer is eventually attained by gradually reducing the distance between the positive and negative intervals. [5] In this procedure, the gap is closed using the average of the positive and negative periods. It is a simple method, but it is time-consuming. Other name of bisection method, the Interval Halving method, the Root-finding method, the Binary search method, and the Bolzano method.

2.1 Bisection Method Algorithm

1. Start
2. Create a function f(x)
3. Choose initial guesses x0 and x1 such that f(x1)f(x2) < 0
4. Choose the acceptable error e.
5. Calculate new approximated root as c = (a+ b)/2
6. Calculate f(a)f(b)
   if f(x0)f(x2) < 0 then x1=x2
   if f(x0)f(x2) > 0 then x0 = x2
   if f(x0)f(x2) = 0 then goto (8)
7. if |f(x2)| > e then repeat (5) otherwise goto (8)
8. Display x2 as root of the equation .
9. Stop

A Example

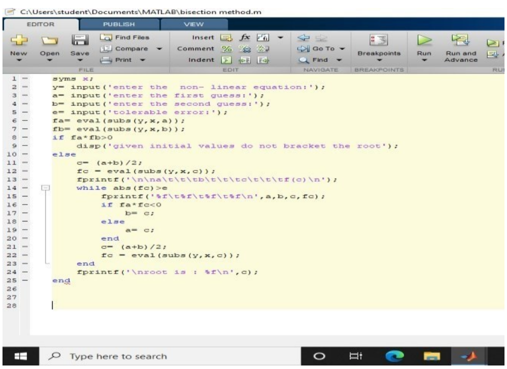

Fig. 1 Bisection Method Example

B. Output

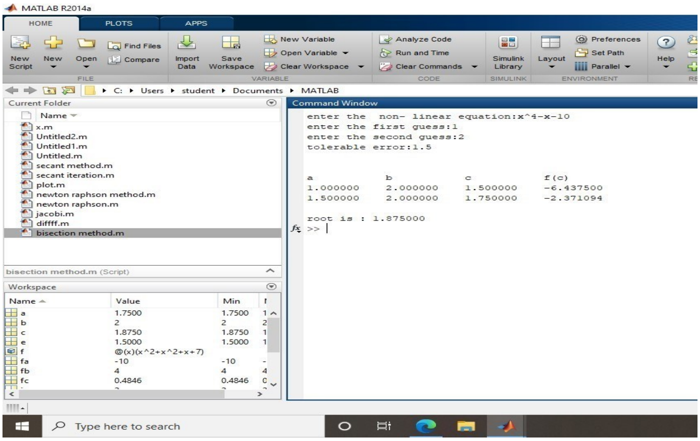

Fig. 2: Bisection Method Output

3. Newton-Raphson Method

Newton's method or Newton–Raphson method is named after Isaac Newton and Joseph Raphson. It is a method for quickly estimating the root of a real-valued function, such as f(x) = 0 or f(x)=0. [6]. It is a powerful and fastest technique for solving equations numerically. In differential calculus, it is based on the simple idea of linear approximation. [8].

A. Newton-Raphson Method Algorithm

1. Start
2. create function as f(x)
3. define derivative of f(x) as g(x)
4. Enter starting guess (x0), tolerable error (e) and maximum iteration (N)
5. Initialize iteration counter i = 1
6. If g(x0) = 0 then print "Mathematical Error" and exit otherwise proceed (7)
7. Calculate: x1 = x0 - f(x0) / g(x0)
8. Increment iteration counter i = i + 1
9. If i >= N then print "Non Convergent" and proceed (12) otherwise proceed (10)
10. If |f(x1)| > e then set x0 = x1 and go to step (6) otherwise to step (11)
11. Print root as x1
12. Stop

A. Example

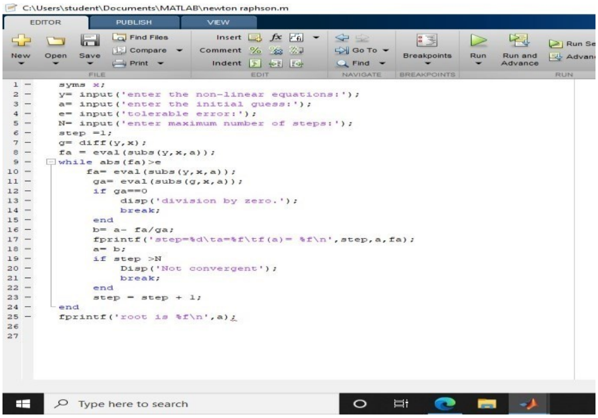

Fig. 3 Newton-Raphson Method Example

B. Output

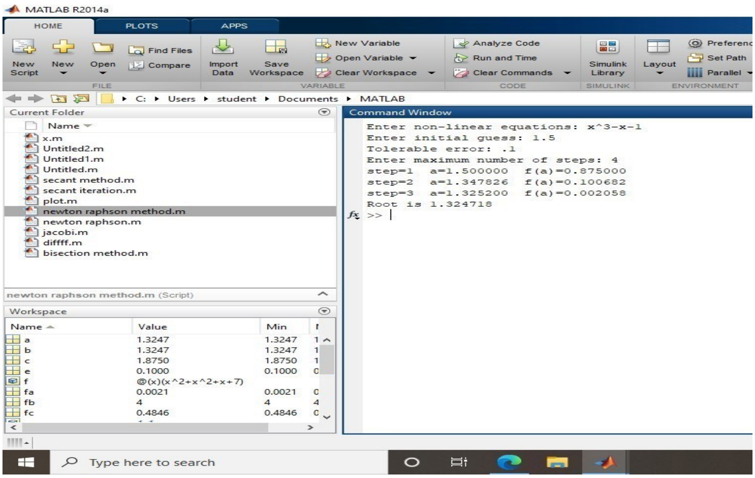

Fig. 4 Newton-Raphson Method Output

4. Secant Method

The Secant method is a root-finding methodology for numerical analysis that makes use of a series of secant line roots to more accurately approximation the root of a function f. Secant Method is open method. It starts with two initial guesses for finding the real root of the non- linear equations. In this method if x0 and x1 are initial guesses then next approximated root x2 is derived by following formula: x2 = x1 - (x1- x0) * f(x1) / (f(x1) - f(x0)) This is an algorithm for Secant method which involves the repetition of the above process i.e., we use x1 and x2 to find x3 and so on until we find the root within desired accuracy.[10]

A. Secant Method Algorithm

1. Start
2. create function as f(x)
3. Enter initial guesses (x0 and x1), tolerable error (tol) and maximum iteration (n)
4. Initialize iteration counter i = 1
5. If f(x0) = f(x1) then prints "Mathematical Error" and proceed (11) otherwise goto (6)
6. Calculate: x2 = x1 - (x1-x0) * f(x1) / (f(x1) - f(x0))
7. Increment iteration counter i = i + 1
8. If i >= N then print "Not Converge" and proceed(11) otherwise proceed (9)
9. If |f(x2)| > e then set x0 = x1, x1 = x2 and proceed with step (5) otherwise goto (10)
10. Print root as x2
11. Stop

B. Example

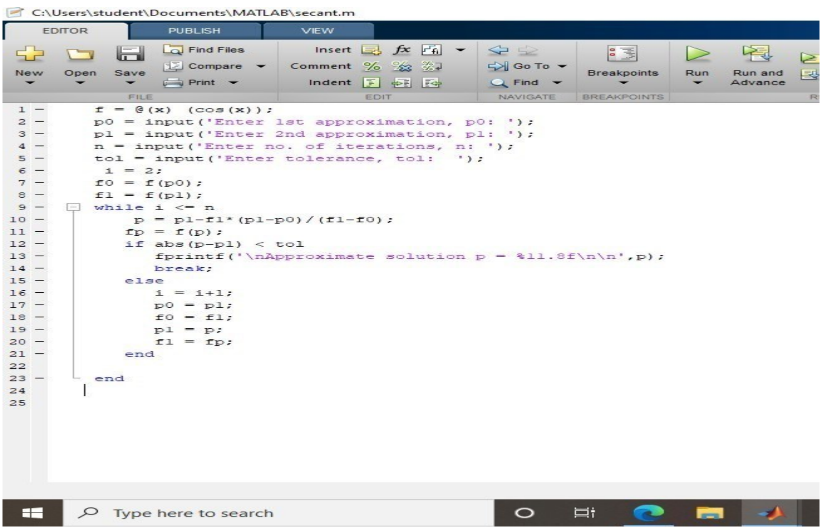

Fig. 5 Secant Method Example

C. Output

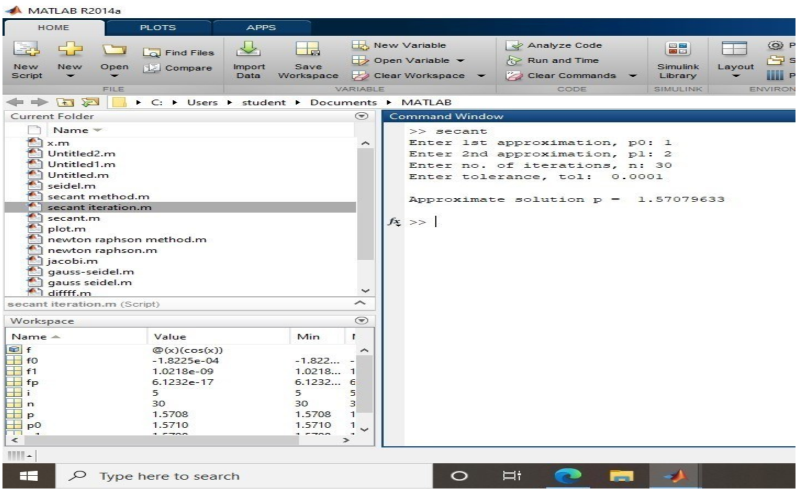

Fig. 6 Secant Method Output

5. Gauss-Jacobi Method

The Jacobi method is an iterative algorithm in numerical linear algebra for finding the solutions of a strictly diagonally dominant system of linear equations. A rough value is entered after solving for each diagonal element. The process is then repeated until convergence isreached. [11].

A. Gauss-Jacobi Method Algorithm

1. Start
2. Arrange given system of linear equations in diagonally dominant form
3. Read tolerable error (tol)
4. Convert the first equation in terms of first variable, second equation in terms of second variable and so on.
5. Set initial guesses for x0, y0, z0 as zero and so on
6. Substitute value of x0, y0, z0 ... from step 5 in equation obtained in step 4 to calculate new values x1, y1, z1 and so on
7. If| x0 - x1| > tol and | y0 - y1| > tol and | z0 - z1| > tol and so on then proceed step 9
8. Set x0=x1, y0=y1, z0=z1 and so on and proceed step 6
9. Print value of x1, y1, z1 and so on
10. Stop

A. Example

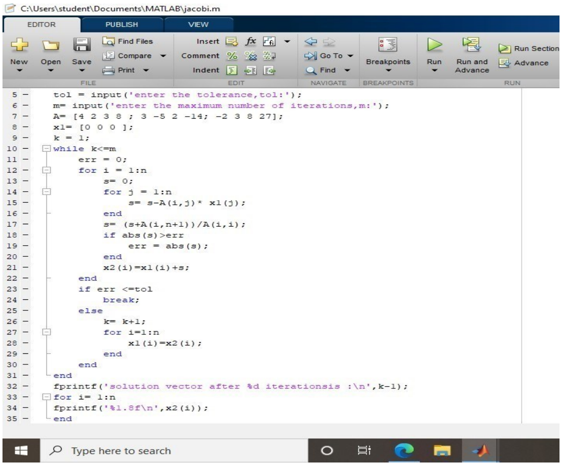

Fig. 7 Gauss Jacobi Method Example

B. Output

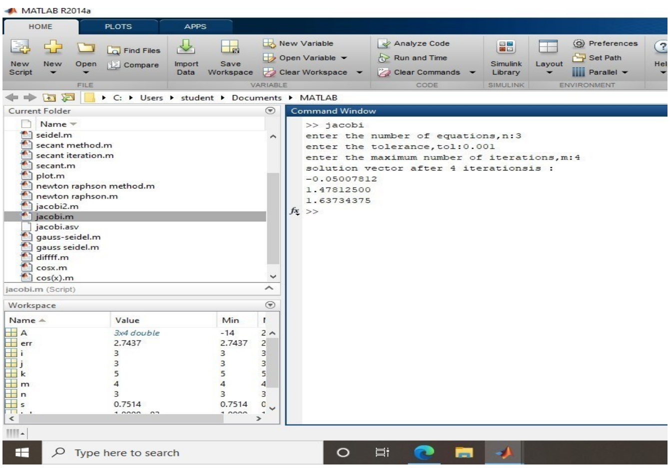

Fig..8 Gauss Jacobi Method Output

6. Gauss-Seidel Method

The Liebman method and the sequential displacement method are other names for the Gauss-Seidel approach. This approach bears the names of the mathematicians Philipp L. Seidel (1821–1896) and Carl Friedrich Gauss (1777–1855). In numerical techniques, an iterative approach is used to solve a set of linear equations. Round-off error can be managed by the user using the Gauss-Seidel Method. Round-off errors can have an impact on the number of methods for elimination. This method is modification of the Jacobi's iteration method. [12].

A. Gauss-Seidel Method Algorithm

1. Start
2. Arrange the system of linear equations in diagonally dominant form
3. Read tolerable error (tol)
4. Convert the first equation in terms of first variable, second equation in terms of second variable and so on.
5. Set initial guesses for y0, z0 and so on as zero.
6. Substitute value of y0, z0 ... from step 5 in first equation obtained from step 4 to calculate new value of x1. Use x1, z0, u0......in second equation obtained from step 4 to calculate new value of y1. Similarly, use x1, y1, u0 ...to find new z1 and so on.
7. If| x0 - x1| > tol and | y0 - y1| > tol and | z0 - z1| > tol and so on then proceed step 9
8. Set x0=x1, y0=y1, z0=z1 and so on and proceed step 6
9. Print value of x1, y1, z1 and so on
10. Stop

A. Example

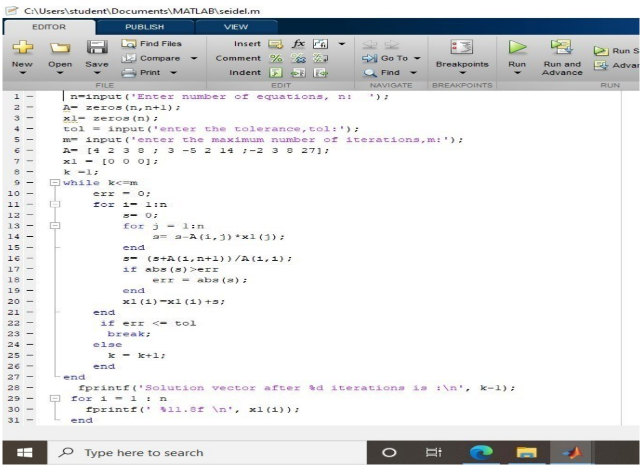

Fig. 9. Gauss Seidel Method Example

B. Output

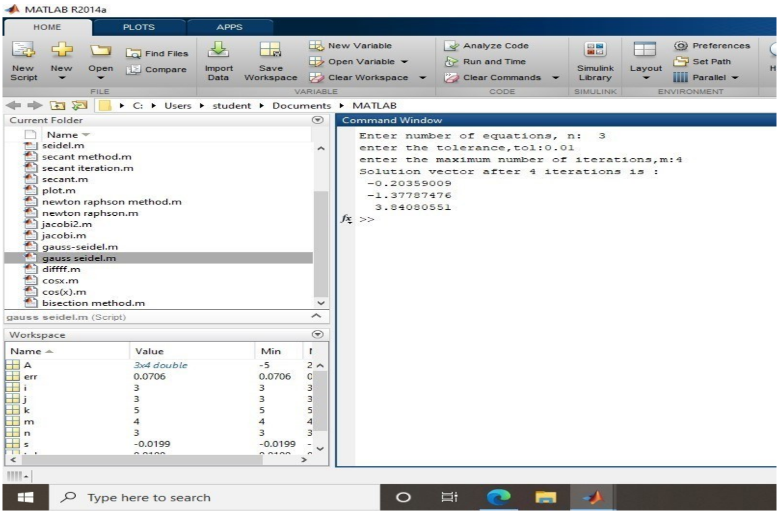

Fig .10 Gauss Seidel Method Output

7. Simpson's 1/3 Rule Method

A numerical approach to find the integral b y dx within the limiting limits a and b in numerical integration is the Simpson's 1/3 rule. The integral of the bounded parabola is computed using Simpson's 1/3 approach, and the approximate integral is then shown. A polynomial of degree two, p(x), or a parabola between the two boundaries a and b, is used to approximate f(x). [14].

1. Begin by defining function f(x)
2. Read the number of sub intervals, the top limit of integration, and the lower limit of integration.
3. Calculate: step size = (upper limit - lower limit)/number of sub interval
4. Set: integration value = f(lower limit) + f(upper limit)
5. Set: i = 1
6. If i > number of sub interval then goto
7. Calculate: k = lower limit + i * h
   If i mod 2 =0 then Integration value = Integration Value + 2* f(k) Otherwise Integration Value = Integration Value + 4 * f(k) End If
8. Increment i by 1 i.e. i = i+1 and go to step 7
9. Compute the Integration value = Integration value * step size/3
10. Show Integration value as the required response
11. Stop

A. Example

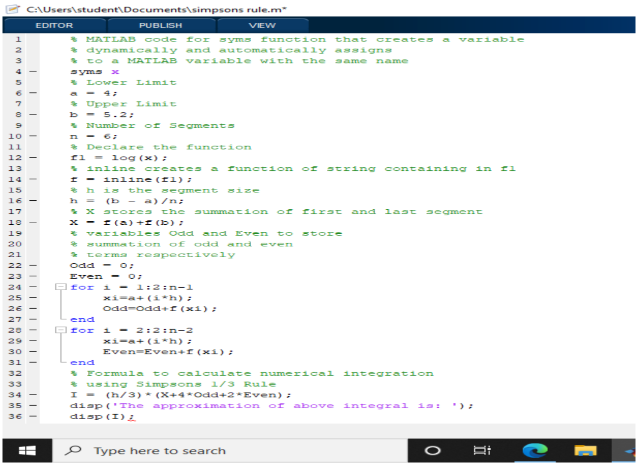

Fig. 11 Simpsons Rule Example

B. Output

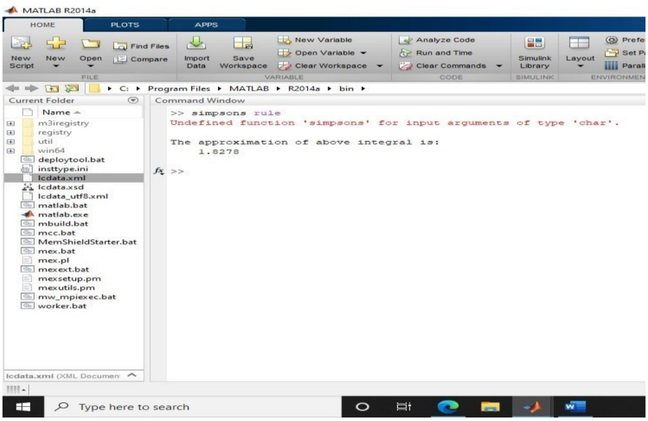

Fig.12 Simpsons Rule Output

8. Conclusion

In this paper, we have discussed variety of numerical methods for finding the solution of ordinary equations with initial conditions by MATLAB Programming. As a result, we can conclude that these numerical methods are useful for obtaining approximate solutions. This work could be expanded in several ways. We have only introduced the fundamental algorithms and MATLAB code in this paper, and we can expand on this by introducing the convergence of the other numerical methods and comparison between different methods by plotting graphs.

References

[1] David Houcque,The MathWorks Inc. MATLAB 7.0 (R14SP2). The MathWorks Inc., 2005.

[2] Attaway, Stormy, " Matlab: A Practical Introduction to Programming and Problem Solving" Canada : Elsevier, Inc., 2009.

[3] Gilat, Amos, "MATLAB An Introduction with Applications", Columbia, Ohio : John Wiley& Sons, Inc., 2011.

[4] Alka Benny, Maria Goldwin, "Numerical Methods Using Matlab", Volume 5, Issue 5, JETIR May 2018.

[5] Ali Wahid Nwry , Hemin Muheddin Kareem , Ribwar Bakhtyar Ibrahim, et al, "Comparison Between Bisection, Newton and Secant Methods for determining the root of the Non-Linear equation using MATLAB", Turkish Journal of Computer and Mathematics Education Vol.12, pp 1115 – 1122, 2021.

[6] Adi Ben-Israel, "A Newton-Raphson Method For The Solution Of Systems Of Equations", Journal Of Mathematical Analysis And Applications 15,pp 243-252, 1966.

[7] Saba Akram, Qurrat ul Ann, "Newton Raphson Method", International Journal of Scientific & Engineering Research, Volume 6, July-2015.

[8] VISHAL V. MEHTRE , SHUBHAM SINGH, "Obtaining Roots of Non-Linear Equation Using Newton Raphson Method", IRE Journals ,Volume 3, DEC 2019

[9] Osama. Y. Ababneh, "New Numerical Methods For Solving Differential Equations", Journal of Advances in Mathematics , vol 16, 2019

[10] Isaac Azure, Golbert Aloliga, Louis Doabil, "Comparative Study of Numerical Methods for Solving Non linear Equations Using Manual Computation", Mathematics Letters, Vol. 5, pp. 41-46, 2019

[11] Yash Bhojwani, Rishab Singh, "Parallelization of Definite Integration", International Research Journal of Engineering and Technology (IRJET), Volume: 06 , Nov 2019

[12] Dr.S.Karunanithi, N.Gajalakshmi, M.Malarvizhi, M.Saileshwari, "A Study on Comparison of Jacobi, GaussSeidel and Sor Methods for the Solution in System of Linear Equations", International Journal of Mathematics Trends and Technology (IJMTT) – Volume 56, April 2018

[13] Riyasdeen S1 , Abbas S2 , Lenin T3. "Illustration of Gauss – Seidel Method Using Matlab", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, pp. 2234-2237, 2019

[14] Md. Jashim Uddin, Mir Md. Moheuddin and Md. Kowsher , " A New Study Of Trapezoidal, Simpson's 1/3 And Simpson's 3/8 Rules Of Numerical Integral Problems ",an international journal(mathSJ), Vol.6, December 2019.

[15] Sharaban Thohura and Azad Rahman, "Comparison Of Numerical Methods For Solving Initial Value Problems For Stiff Differential Equations", Ganit J. Bangladesh Math. Soc. 30, pp 122-132, 2010.