DATA ANALYTICS AND APPLIED MATHEMATICS
E-ISSN: 2773-4854
VOLUME 6, ISSUE 1, 2025, 23 – 35
DOI: https://doi.org/10.15282/daam.v6i1.11307

UNIVERSITI MALAYSIA PAHANG AL-SULTAN ABDULLAH

RESEARCH ARTICLE

Efficiency of Ridders' method in solving nonlinear equations using Scilab programming

Annie Gorgey*, Farah Adilla Azim, Zul Hafiezy Zulkifly, Haslinah Hassim

Department of Mathematics, Faculty of Science and Mathematics, Sultan Idris Education University, 35900 Tanjong Malim, Perak, Malaysia

ABSTRACT - The study aims to compare the efficiency of numerical methods such as Bisection, Secant, Newton-Raphson, Ridders and Halley's methods in solving nonlinear scalar equations. The research provides numerical experiments on the efficiency and accuracy of the methods by focusing on the number of iterations, accuracy, and computational time. Based on the numerical results, Ridder's method outperforms other methods in terms of accuracy and efficiency for all the nonlinear problems. Although the Secant method did perform well for problems involving polynomial function and the Bisection method did perform well for problems involving exponential function, the method is not as efficient as Ridder's method in terms of computational time. Newton-Raphson method although gives quadratic convergence the method has slightly higher number of iterations than the Ridder's and Halley's methods. Hence, the research underscores the significance of numerical methods in solving nonlinear equations by using an open-source programming language which is Scilab.

ARTICLE HISTORY
Received : 20th Sept 2024
Revised : 15th Feb 2025
Accepted : 10th Mac 2025
Published : 31st Mac 2025

KEYWORDS
Ridders' method
Bisection
Secant
Newton-Raphson
Two step Halley's method
CPU time

1. INTRODUCTION

Numerical methods are techniques in which mathematical problem are formulated to be solved using mathematical operations and computing device. Although there are many types of numerical methods, they have one common feature and often these numerical methods involve large numbers that will lead to relatively long arithmetic calculations. Numerical methods also play a crucial role in solving nonlinear equations when analytical solutions are difficult or impossible to obtain. These equations arise in various fields of science, engineering, and mathematics. To tackle such problems, a programming software called Scilab, a powerful numerical computing environment, provides a wide range of tools and functions for implementing numerical methods.

In line with today's modernization trend, it is not surprising that the development of digital computers is now becoming faster and more efficient so that the role of the subjects in numerical methods in solving engineering problems has increased dramatically in recent years. Nowadays, computers and numerical methods have various alternatives to solve complex calculations. As you know, by using the existing facilities on the computer to get solutions or problems directly, it can lead to the calculation of problems or questions without having to use methods to simplify assumptions or time intensive techniques.

Scilab is a programming language associated with a wide variety of numerical algorithms involving many aspects, especially scientific computing problems. This software can also be used for statistical analysis such as estimating the parameters in the Gompertzian curve using the Gauss-Newton method. There are many features in this software and rapidly the features in Scilab are increasing and expanding to cover many areas of scientific computing. Among the topics suitable for using scilab software are matrices, polynomials, linear equations, ordinary differential equations (ODE) and statistics [1].

Scilab is an open-source software that offers a user-friendly interface for numerical analysis and scientific computations. It provides a vast collection of functions for solving nonlinear equations using various numerical methods. Some popular methods supported by Scilab include, Bisection method, Newton-Raphson method, Secant method, Halley's and Ridder's Methods.

The methods stated can be used to solve nonlinear equations. Some of them are being taught by the teachers or lecturer at the school or in the university. This new methods in solving non-linear equations may provide better information or knowledge from the viewpoint of the accuracy of some iterative methods. In this paper, a numerical method for the nonlinear equations' solution will be presented, as well as to compare which method has the lowest number of iterations by using Scilab Programming.

To use these methods in Scilab programming, we can define the nonlinear function $f(x)$ select an appropriate method and implement it using Scilab's programming features. Scilab provides functions like 'fsolve' and 'roots' for solving nonlinear equations, which internally use these methods. Scilab also provides visualization capabilities to plot the function and observe the convergence of different methods. Thus, Scilab is a powerful tool for implementing numerical methods to solve nonlinear equations. Its extensive set of functions and programming features enable efficient and accurate

***CORRESPONDING AUTHOR | Annie Gorgey** | ✉ annie_gorgey@fsmt.upsi.edu.my

solutions to a wide range of problems in scientific and engineering domains. There is also other software that can be used to solve the numerical methods, but most existing methods is either expensive to purchase or difficult to implement and not worth considering when solving a simple problem that does not require many iterations to obtain the approximation.

A simple mathematics problem can be linear or nonlinear equations or scalar and system of equations. Nonlinear equations are equations that involve nonlinear functions of the unknown variables. The general form of a nonlinear equations is $f(x) = 0$, where $f(x)$ is a nonlinear function and x represents the unknown variables. The goal is to find the values of x that satisfy the equation. Nonlinear equations are equations that also appeared as curved lines when the function is being graph [2]. Nonlinear equation is chosen for this article since in most cases the exact solution is hard to find, and the solution requires accurate numerical method to find the solution.

In addition to that, many the nonlinear equations are used in engineering where solving equations related to heat transfer, fluid dynamics, and electrical circuit analysis where robustness is crucial. There are also applications involving physics and environmental science. In Physics the equation involved in solving quantum mechanics and other areas where functions might have complex behaviours and require reliable convergence [3].

This article provides numerical experiments on several nonlinear scalar problems involving trigonometric, exponential, logarithm and polynomial functions solved by five methods such as the Bisection, Secant, Newton Raphson, Ridder's and Halley's method. The detailed explanation about these methods is given in Section 2. The comparison of these methods is given in terms of computational time to determine its efficiency and absolute errors for accuracy. The results are presented in Section 3.

2. NUMERICAL METHODS

In this research, we consider five types of numerical methods which are, Bisection, Secant, Newton-Raphson, Ridders' and two-step modified Halley's method (HM) methods.

2.1 Bisection Method

One of the simplest algorithms for finding algebraic equations is the bisection method. This method can be used for any continuous function f on in the domain $[a, b]$. According to intermediate Value Theorem, if $f(a).f(b)$ has opposite signs, there is at least one root in this range [4]. The pseudo code for Bisection method is given as follows:

Start

Step 1: Define function $f(x)$

Step 2: Define initial value and parameters
- i. Set the endpoints: a and b.
- ii. Set the Tolerance, epsilon e.
- iii. Starting iteration, i

Step 3: Compute the iteration by Bisection

while $i > 0$
  $c = \frac{a+b}{2}$
if $f(a).f(c) < 0$ then
  $a = a$
  $b = c$
else
  $a = c$
  $b = b$
end
  $a = c$

Step 4: Convergence test

if abs$(f(c)) < e$
  break
update $i = i + 1$

Step 5: Print root as c

Stop

The bisection method is a simple and robust technique for finding the root of a function. It works by repeatedly dividing an interval in half and selecting the subinterval in which the root must lie. The method is guaranteed to converge

to a root if the function gives opposite sign over the interval. This method can only locate the root if the endpoints chosen are within the root. One of the disadvantages of this method is that only one root can be located. A smaller interval is recommended to find the root.

Atkinson [2] provides a detailed and rigorous introduction to numerical analysis, including the bisection method. This method is praised for its robustness and simplicity, making it a reliable choice for solving nonlinear equations. The text elaborates on the convergence properties of the bisection method and its implementation, emphasizing its guaranteed convergence when the function is continuous, and changes sign over the interval. The latest reference to this method is given by Osman et. al [4] where the use the bisection method to locate the method stops if one of the midpoints happens to coincide with the root. Other applications are given in [3].

2.2 Secant Method

The secant method is a root-finding algorithm that uses a sequence of secant lines to approximate the root. It is similar to the Newton-Raphson method but does not require the calculation of derivatives, making it useful when the derivative is difficult to obtain. The secant method is an iterative numerical approach for finding the roots of a real-valued function. It starts with two initial guesses x_0 and x_1, and at each iteration, it constructs a secant line between the function values at these points to approximate the root. The method updates the guesses based on the intersection of the secant line with the x-axis using the formula:

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}. \quad (1)$$

This process continues until a satisfactory approximation of the root is achieved or a predetermined number of iterations is reached. Burden and Faires [4] discuss the secant method as a practical alternative to Newton-Raphson when derivative calculations are cumbersome or impractical. The secant method, which approximates the derivative using finite differences, is presented as a bridge between the bisection method's simplicity and Newton-Raphson's speed. The text includes examples, convergence analysis, and potential pitfalls of the method. The pseudo code for Secant method is given as follows:

Start

Step 1: Define function $f(x)$

Step 2: Define initial value and parameters

i. Set $x_0 = a$ and $x_1 = b$
ii. Set tolerance, epsilon e.
iii. Set, i=1

Step 3: Compute the iteration by Secant

while $i > 0$

$$x = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$$

update $x_0 = b$ and $x_1 = x$

Step 4: Convergence test

if abs$(f(x)) < e$
break
end
update $i = i + 1$

Step 5: Print root as x

Stop

A comparative study of nonlinear root finding using improvised secant methods was given in [6] on which they have two algorithms based on the Secant method which are the exponential method, and three-point Secant method that were used to compare with the Secant method to evaluate the roots for nonlinear equations. Their findings show that the three-point Secant method has the least number of iterations than the Secant method and is exponential and concluded that the three-point Secant method is the best iterative method since the method converged to the roots faster.

2.3 Newton-Raphson Method

The Newton-Raphson method is an efficient root-finding algorithm that uses the derivative of the function to iteratively improve an initial guess. It converges quickly when the initial guess is close to the actual root, but it may fail to converge if the initial guess is not well-chosen or if the function is not well-behaved. The Newton-Raphson method, also known as Newton's method, is an iterative numerical technique for finding the roots of a real-valued function. It requires an initial guess x_0 and iteratively refines this guess using the formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{2}$$

Here, $f(x_n)$ represents the function value at the current guess, and $f'(x_n)$ is the derivative of the function at that point. The method exploits the tangent line at the current guess to estimate the root, making it particularly effective for functions where the derivative is readily available or computationally feasible to obtain. The process continues until a satisfactory approximation is reached or a specified number of iterations are performed. The pseudo code for Newton method is given as follows:

Start

Step 1: Define function $f(x)$ and $f'(x)$

Step 2: Define initial value and parameters

i. Set initial guess, x_0
ii. Set tolerance, epsilon e
iii. Set, i=1

Step 3: Compute the iteration by Newton

while $i > 0$
$x = x_0 - \frac{f(x_0)}{f'(x_0)}$.
update $x_0 = x$

Step 4: Convergence test

if abs$(f(x)) < e$
break
end
update $i = i + 1$

Step 5: Print root as x

Stop

Chapra and Canale's comprehensive guide to numerical methods dedicates substantial coverage to the Newton-Raphson method, highlighting its efficiency and rapid convergence. The method is especially useful for engineering applications where speed and accuracy are critical. The book provides detailed algorithms, convergence criteria, and examples demonstrating the method's application to various types of nonlinear equations, as well as strategies to handle potential divergence issues [7].

2.4 Ridders' Method

Ridders' method is a numerical algorithm for finding the roots of a real-valued function. It is an improvement over the bisection method and belongs to the family of root-finding methods that do not require knowledge of the derivative. Ridders' method combines the bisection technique with linear interpolation to provide more efficient and robust convergence. The idea is given by Ridder [8] which is based on false position method using exponential function to find the root of a function. Although the method is quadratic convergence which is similar to the Newton method, the method requires a complication evaluation of the function. The function has to be evaluated twice for every step and therefore it is interesting the performance of this method when compared with the famous Newton-Raphson method. The pseudo code for Ridder's method is given next.

From the given pseudo code, the first step is to consider the midpoint $c = \frac{(a+b)}{2}$ and find $f(c)$. Next, the unique exponential function is factored out of f to make the three points $(a, f(a)), (c, f(c)), and (b, f(b))$ colinear. This corresponds to finding an e^Q which is defined as x in the given pseudo code.

```
Start

Step 1: Define function f(x)

Step 2: Define initial value and parameters

        i.    Set the interval  xb = [a, b]
        ii.   Set tolerance, epsilon e
        iii.  Set, i=1

Step 3: Test to find the proper bracketing
        yb = [f(a), f(b)]
        if sum(sign(yb))!=0
        display error message
        end

Step 4: Compute the iteration by Ridders
        while i > 0
          xc= (a+b)/2
        Set y = f(c)
          x = (f(c)+sign[f(b)]√(f(c)^2−f(a)f(b)))/f(b)
        Set yn = f(x)
          if y.yn < 0
              xb = [xc, xc]
              yb = [y, yn]
          else
              replace=sign(yn)==sign(yb);
              xb(replace)=x;
              yb(replace)=y;
          end

Step 5: Convergence test
        if y == 0||( bounds(2) - bounds(1) < e)
             break
        end
        update i = i + 1

Step 6: Print root as x

Stop
```

The root is approximated when $f(a) - 2f(c)e^{Q} + f(b)e^{2Q} = 0$. This equation is equivalent to enforcing that the centred-difference stencil for the second derivative $f(a), f(c)e^{Q}, f(b)e^{2Q}$ is zero. This equation is a quadratic in e^{Q} and has the solution as given in [9].

$$x = e^{Q} = \frac{f(c)+\text{sign}[f(b)]\sqrt{f(c)^2-f(a)f(b)}}{f(b)}. \tag{3}$$

Since, $f(a)$ and $f(\text{b})$ have opposite sign, the square root will always be real. Once this is determined, a false position step is performed on the function with the exponential factored out. Performing linear interpolation yields the false position step of,

$$d = c + (c - a)\frac{\text{sign}[f(a)-f(b)]f(c)}{\sqrt{f(c)^2-f(a)(b)}}. \tag{4}$$

Based on the sign of $f(d)$ the bracketed region is replaced with either (a, d) if $f(d) < 0$, or (d, b) if $f(d) > 0$. However, a further optimization is possible when the sign $f(c)$ differs from $f(d)$. In that case, the bracketed region can be reduced further to (c, d) or (d, c) depending on the ordering of c and d.

2.5 Two-step modified Halley's method (HM)

The two-step modified Halley's method (HM) is proposed by Noor et al. [10] and known as Algorithm 2.4 in their article. This method does not require the second derivative. Important characteristic of this method is that per iteration

they require two evaluations of the function and one of its first derivatives. The efficiency of this class of methods is better than that of the well-known other methods involving the second-order derivative of the function:

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{5}$$

$$x_{n+1} = y_n - \frac{2f(x_n)f(y_n)f'(y_n)}{2f(x_n)f'^2(y_n) - f'^2(x_n)f(y_n) + f'(x_n)f'(y_n)f(y_n)}. \tag{6}$$

This method is chosen in this article because it is given in [10] that the method exhibits fifth order convergence and is very efficient. The pseudo code for two-step modified Halley method is given as follows:

Start

Step 1: Define function $f(x)$ and $f'(x)$

Step 2: Define initial value and parameters

i. Set initial guess, x_0
ii. Set tolerance, epsilon e
iii. Set, i=1

Step 3: Compute the iteration by Newton

while $i > 0$

$$y_n = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x = y_n - \frac{2f(x_0)f(y_n)f'(y_n)}{2f(x_0)f'^2(y_n) - f'^2(x_0)f(y_n) + f'(x_0)f'(y_n)f(y_n)}$$

update $x_0 = x$

Step 4: Convergence test

if abs$(f(x)) < e$
break
end
update $i = i + 1$

Step 5: Print root as x

Stop

3. IMPLEMENTATION AND ALGORITHM

The four numerical methods are implemented in Scilab programming language version Scilab 2024.1.0 (64 bit) on Windows 11 using ThinkPad Laptop with Intel(R) Core (TM) i7-8565U CPU @ 1.80GHz 1.99 GHz. The performance of the numerical methods is assessed based on the key criteria, including:

- Iterations: The number of iterations needed for the equations to converged, reflecting the method's convergence rate.
- Accuracy: The closeness of the obtained root to the exact root for the equation towards each of the numerical methods, emphasizing the precision of the numerical solution. The accuracy of the method is measured from the Absolute Error.
- Computational Efficiency: The number of CPU time needed for the equations to converged, reflecting the method's convergence rate.

These criteria collectively provide a robust foundation for evaluating the efficacy of numerical methods in handling the nonlinear equations. The provided description on the Scilab code encompasses several key components as given in Table 1.

Table 1. Description of the Scilab key components used in the numerical experiments

Key Components	Description
Function definition, $f(x)$	• The function of $f(x)$ is defined to find the root of the function. • It takes a value of x as input and computes the difference between each mathematical expression. • The function $f(x)$ is defined in a function file
Initial of Parameters	• The initial parameter consists of the followings: i. Tolerance, *eps* ($\varepsilon = 1e^{-15}$). Any stringent value can be used. The more stringent value is used, the more accurate the approximation and closer to the exact solution. Recommended tolerance is $1e^{-5}$ [4]. ii. Initial interval or boundary for the root-finding algorithm [a, b] (the interval where the root is expected) iii. Initial guess or starting value x_0 (any choice of starting value) iv. The maximum number of iterations: 50
Iterative Methods	• The core of the code lies in the implementation of the study iterative methods. The iterative methods are defined in a function file. • The while loop iterates through a maximum number of iterations. • Inside the loop, it calculates the estimated root using each method formula based on each function and initial parameters.

Convergence Test	For the true statement, if the absolute value is less than the defined epsilon (`eps`) or tolerance, the loop breaks, indicating the convergence.
	`if (abs(function < eps) then`
	`break`
	`end`
	Whenever this statement is satisfied, the iteration will break and print the last number of iterations and the desired root.

Central Processing Unit (CPU) Time Measurement	• The '`tic ()`' and '`toc ()`' functions are used to measure the CPU time taken by each method.
	• The '`timer ()`' function is used to display the total CPU time taken by the script.
	• The '`format (10)`' function sets the display format for numerical values.
	`tic();`
	`[root_method,iter1] = method_name(f, a, b, eps);`
	`time_method = toc();`

Absolute Error Calculation	• The `fsolve` function is used to find the exact root (z) of the function.
	• The absolute error is calculated as the absolute difference between the computed root (root_method) and the exact root (z).
	`z=fsolve(method_name,f);`
	`error =abs(root_method-z);`

Printing Headers, Result and Output	• These lines print headers for the table of results that will be displayed during each iteration using the mprintf statement. • For the result, it prints the estimated root, and the number of iterations required to achieve the desired accuracy. • The graph is produced using scilab plot.

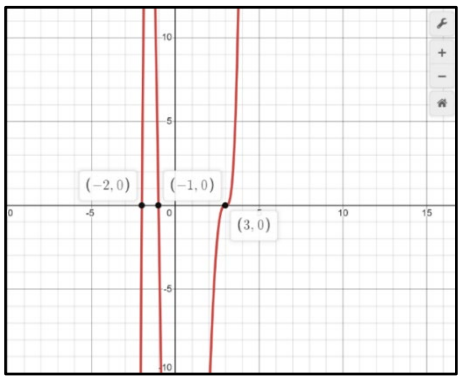
4. NUMERICAL RESULTS AND DISCUSSION

There are six types of nonlinear problems considered in this article. Each problems have different types of functions. We have considered problems with the following types:

a) Equation 1: Polynomial up to degree 5
b) Equation 2: Logarithmic function
c) Equation 3: Sine function and algebraic function
d) Equation 4: Exponential function
e) Equation 5: Combinations of exponential and trigonometric functions.
f) Equation 6: Combination of a quadratic term, a sine function with a small argument, and a constant shift.

The nonlinear problems considered in this article include is taken from [9-11].

a) Equation 1: $f(x) = (x+2)(x+1)(x-3)^3$ within [-2.5,3]
Equation 1 is a polynomial function of degree 5 given in [11]. This polynomial equation is a quintic function, and its graph in the Cartesian plane would be a smooth curve that may have up to 4 turning points. These polynomial equations are used in various fields, including physics, engineering, economics, and computer science, to model and analyse real-world phenomena. As example, in control systems, polynomial equations can describe the behaviour of dynamic system while in physics, polynomial equations can represent certain physical quantities or relationship.



Root of the function, $x = -2, -1$ and $x = 3$

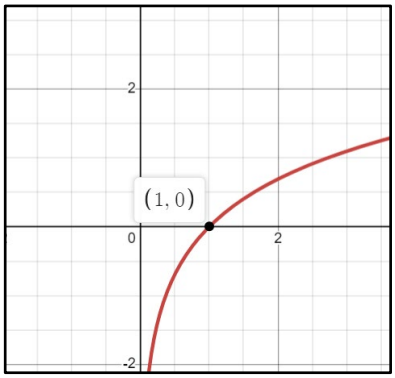
Figure 1. The graph of the function, $f(x) = (x+2)(x+1)(x-3)^3$ within [2, 3.5]

Table 2. A comparison of different numerical methods for equation 1

Method	Approximate Root	Iteration No.	Absolute Error	Computational (CPU) Time (seconds)
Bisection	3.000001907348632812500000	18	1.907349e-06	0.001046
Secant	3.000002982410742813357273 9	41	2.982411e-06	0.002166
Newton-Raphson	2.999996430323806251294627 1	42	3.569676e-06	0.031111
Ridders'	2.999998229603207722249180 7	15	1.770397e-06	0.000497
HM	2.999996430901822108694432 2	41	3.569098e-06	0.049834

Based on Figure 1, the root of the function at the interval [2, 3.5] is $x = 3$. All the numerical methods converged to the exact root. Table 2 shows that the most accurate and efficient method is Ridder's method that converged in 0.000497 seconds within 15 iterations. Bisection method only needs additional 3 more iterations to converge to the root. Both Bisection and Riders' methods have almost similar accuracy.

b) Equation 2: $f(x) = \ln(x)$ within [0, 2]
Equation 2 is involving the natural logarithm function. The functions have variety of applications such as in growth and decay model, signal processing, control system, compound interest and many more.



Root of the function, $x = 1$

Figure 2. The graph of the function, $f(x) = \ln(x)$ within [0, 2]

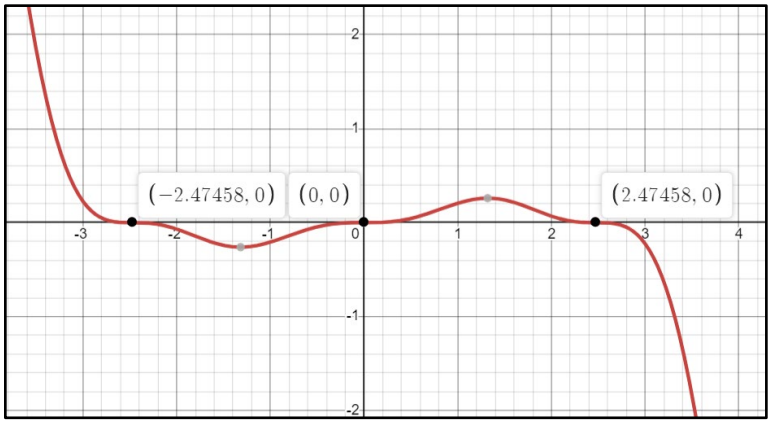
Table 3. A comparison of different numerical methods for equation 2

Method	Approximate Root	Iteration No.	Absolute Error	Computational (CPU) Time (seconds)
Bisection	1.000000000000000008881784197	49	8.881784e-16	0.001438
Secant	1.000000000000000000000000	8	0.000000e+00	0.000300
Newton-Raphson	1.000000000000000000000000	5	0.000000e+00	0.006050
Ridders'	1.000000000000000000000000	5	0.000000e+00	0.000235
HM	1.000000000000000000000000	4	0.000000e+00	0.006502

Based on Figure 2, the root of the function at the interval [2, 3.5] is $x = 1$. All the numerical methods converged to the exact root. Table 3 shows that the most the accurate and efficient method is Ridder's method that converged in 0.000235 seconds within 5 iterations. Secant method came second with 0.0003 seconds with 8 iterations followed by HM and Newton-Raphson methods. Bisection method has the smallest absolute errors but requires 49 iterations to converge. In an article published in *Matrix Science Mathematic* (MSMK) (2022) also showed that for the similar equation, Ridder's method did outperform Bisection and Regular Falsi methods [9], however that article use different tolerance therefore the number of iterations is much lesser than Table 3 given in this article.

c) Equation 3: $f(x) = \left(\sin(x) - \frac{x}{4}\right)^3$ within [-1, 2]

Equation 3 is a mathematical expression involving the sine function and algebraic operations, raising the result to the power of 3. The sine function is periodic, oscillating between -1 and 1 for all real values of *x*. the sine function is combined with an algebraic expression $\frac{x}{4}$ inside parentheses. This equation can be applied in diverse fields. In physics and engineering to describe oscillatory phenomena, in economic models to represent cyclical economic patterns, in mechanical engineering used to describe vibrations and oscillations in mechanical systems and any other else.



Root of the function, $x = -2.47458, 0$ and $x = 2.48458$

Figure 3. The graph of the function, $f(x) = \left(\sin(x) - \frac{x}{4}\right)^3$ within [-1, 2]

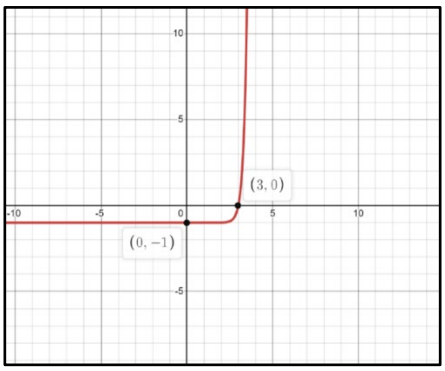
Table 4. A comparison of different numerical methods for equation 3

Method	Approximate Root	Iteration No.	Absolute Error	Computational (CPU) Time (seconds)
Bisection	0.000007629394531250000000	16	7.629395e-06	0.000371
Secant	0.000010197013044329843355 0	37	1.019701e-05	0.000949
Newton-Raphson	0.000011864566780408380064 1	26	1.186457e-05	0.016484
Ridders'	0.000008254995846289566810 7	13	8.254996e-06	0.000466
H	0.000011440832264975917424 0	24	1.144083e-05	0.029461

Based on Figure 3, the root of the function at the interval [-0.5, 1] is $x = 0$. All the numerical methods converged to the exact root. Table 4 shows the accurate and efficient method is Ridder's method that converged in 0.000466 seconds within 13 iterations. Bisection method came second with 0.000371 seconds with 16 iterations followed by the Secant method.

d) Equation 4: $f(x) = e^{(x-3)^5} - 1$ within [2,4]

Equation 4 is an exponential function characterized by a quintic degree-5 power term. The term $(x - 3)$ inside the exponent introduces a horizontal translation, shifting the graph 3 units to the right. The subtraction of 1 at the end of the expression results in a vertical shift downward by one unit. For larger values of *x,* the exponential growth dominates, and the function increases rapidly. The equation has potential applications in various scientific, engineering, and mathematical contexts. As an example, in financial modelling the function represents compound interest or growth, while in data analysis, the function is employed for transforming data or fitting curves. There are many other else applications of this equation that depend on the characteristics of the system or phenomenon being modelled and adjustments to the parameters.



Root of the function, $x = 3$

Figure 4. The graph of the function, $f(x) = e^{(x-3)^5} - 1$ within [2,4]

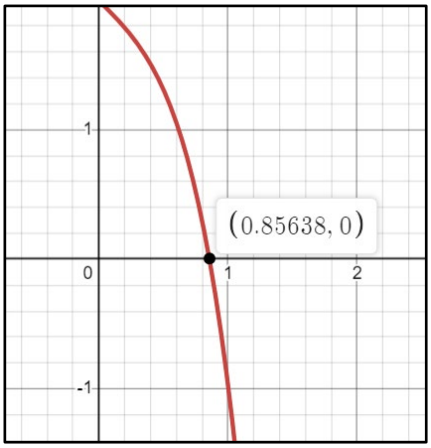
Table 5. A comparison of different numerical methods for equation 4

Method	Approximate Root	Iteration No.	Absolute Error	Computational (CPU) Time (seconds)
Bisection	3.000000000000000000000000	1	0.000000e+00	0.000088
Secant	2.999119099929895647484642	42	0.000000e+00	0.001148
Newton-Raphson	2.999000380458162862140625 3	30	0.000000e+00	0.021435
Ridders'	3.000000000000000000000000,	1	0.000000e+00	0.000096
HM	2.999059671117248804250721 2	27	0.000000e+00	0.033350

Based on Figure 4, the root of the function at the interval [2, 4] is $x = 3$. All the numerical methods converged to the exact root. Table 5 shows that the most accurate and efficient method is Bisection method that converged in 0.000096 seconds within 1 iteration. Ridders' method came second with 0.000096 seconds with also 1 iteration almost similar accuracy with the Bisection method. HM method required more computational time to converge to the root in 27 iterations.

e) Equation 5: $f(x) = x\sin(x) - x^2e^x + e^{-x} + \cos(x)$ within [-1,1]

Equation 5 is a complex, non-linear function involving both exponential growth/decay and oscillations. Terms like $x\sin(x)$ and $\cos(x)$ appear in models of oscillatory systems, such as springs, pendulums, and electrical circuits. In mechanical systems, oscillatory motion might be driven by forces proportional to both x and the sine of x, with damping or forcing functions. Exponentially growing terms, such as x^2e^x, may appear in unstable control systems, where the behavior of a system (like a robot arm or an aircraft) becomes increasingly uncontrolled unless corrective measures are applied. A combination of trigonometric and exponential functions could also describe damped oscillations. Such models are used in mechanical systems with friction, electrical circuits with resistance, and other contexts where energy is gradually lost [12-13].



Root of the function, $x = 0.85638$

Figure 5: The graph of the function, $f(x) = x\sin(x) - x^2e^x + e^{-x} + \cos(x)$ within [-1,1]

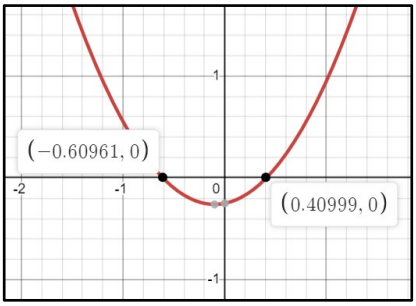
Table 6. A comparison of different numerical methods for Equation 5

Method	Approximate Root	Iteration No.	Absolute Error	Computational (CPU) Time (seconds)
Bisection	0.85637991380523037499017 39	53	0.000000e+00	0.002424
Secant	0.85637991380523037499017 39	8	0.000000e+00	0.000392
Newton-Raphson	0.85637991380523037499017 39	9	0.000000e+00	0.007680
Ridders'	0.85637991380523037499017 39	4	0.000000e+00	0.000223
HM	0.85637991380523037499017 39	7	0.000000e+00	0.009416

Based on Figure 5, the root of the function at the interval [-1, 1] is $x = 0.85638$. All the numerical methods converged to the exact root. Table 6 shows that the most accurate and efficient method is Ridders' method that converged in 0.000223 seconds within 4 iterations. Secant method came second with 0.000392 seconds with 8 iterations. According to Shnyar Karim etc al. [9] their numerical results for the similar problem with tolerance 1e-4, Bisection method converges in 15 iterations and Ridder's method in 4 iterations. Although they comparison are not fair to be made, the article did not give any other details except stating that their new proposed method can converge in 2 iterations. We can also see that HM requires less iteration than Newton-Raphson but more computational time. This result is accepted because HM has double function evaluation computation as given in equation (6) since the given equation 5 has the combinations of exponential and trigonometric functions.

f) Equation 6: $f(x) = x^2 + \sin\left(\frac{x}{5}\right) - \frac{1}{4}$ within [0,1]

This function combines a quadratic term, a sine function with a small argument, and a constant shift. The function is nonlinear due to the combination of the quadratic term and the sinusoidal component. For small values of x, the sine term dominates, causing mild oscillations around the quadratic curve. As $|x|$ increases, the x^2 term dominates, and the function grows rapidly. There will be local extrema (local minima and maxima) due to the oscillations introduced by the sine term. Nonlinear functions like this one can appear in optimization scenarios, where the goal is to find the minimum or maximum values. The oscillatory component adds local minima and maxima, making it suitable for studying more complex optimization problems involving multiple solutions [14]. Similar equation is used by Noor etc. al. in [10] where they used HM method.



Root of the function, $x = -0.60961$ and $x = 0.40999$

Figure 6. The graph of the function, $f(x) = x^2 + \sin\left(\frac{x}{5}\right) - \frac{1}{4}$ within [0, 1]

Table 7. A comparison of different numerical methods for Equation 6

Method	Approximate Root	Iteration No.	Absolute Error	Computational (CPU) Time (seconds)
Bisection	0.40999201798913631478171 74	48	8.326673e-16	0.001313
Secant	0.40999201798913714744898 58	8	0.000000e+00	0.000323
Newton-Raphson	0.40999201798913714744898 58	7	0.000000e+00	0.006193
Ridders'	0.40999201798913720296013 71	5	5.551115e-17	0.000204
HM	0.40999201798913720296013 71	5	5.551115e-17	0.006502

Based on Figure 6, the root of the function at the interval [0, 1] is $x = 0.40999$. All the numerical methods converged to the exact root. Table 7 shows that the most accurate and efficient method is Ridders' method that converged in 0.000204 seconds within 5 iterations. Noor etc. al. in [10] proved that HM which is known as Algorithm 2.4 is the best method if compared with the other five different methods including the Newton Raphson method, the opposite result is obtained in this case. As we can see in Table 7, the Ridders' method gave similar results with the HM method but perform much faster than HM method even for a very stringent tolerance. On the other hand, interestingly we can observe that although the Bisection method is faster than Newton-Raphson method, it requires more iterations than Newton-Raphson because the method guaranteed convergence as long as the function is a continuous function and only requires evaluating the function at the to test whether the root exist in the given interval. This also due to the simplicity of the given function.

5. CONCLUSIONS

The convergence rates of the numerical root-finding methods, evaluated through the lenses of the number of iterations, absolute error, and computational time, offer insights into their relative efficiency. Ridders' method, featuring higher-than-linear convergence, leads the pack by swiftly reducing the error in fewer iteration compared to the other method. Ridders' method excels in computational efficiency, making it a favourable choice when a balance between speed and robustness is sought. The bisection method follows, when its linear convergence necessitating a predictable number of iterations for each step but at slower pace than Ridders' method. The Secant method, with super linear convergence, sometimes outpaces the Bisection method in both iterations and computational time. Newton-Raphson, known for quadratic convergence, making it effective when the initial guess is close to the root. The reliability of the Ridders method is very high, and it is more accurate than Bisection, Secant and Newton-Raphson. HM method on the other hand could perform faster if the function involved is not a complicated function since the method involves more than one function evaluation.

Based on the result of the computations above, most of the nonlinear problems, Ridders' method converges faster than Bisection, Secant, Newton-Raphson and HM method. The article proved that even with the use of Scilab programming language and with a very stringent tolerance all the methods can converge to the exact solution. There are many other programming languages that can be used to solve the numerical method such as Maple, Phyton and Matlab as given in [9, 10, 15]. Scilab programming is an open-source software, and it is recommended to solve simple and non-complex mathematical problems.

ACKNOWLEDGEMENTS

Institution(s)
The authors would like to express gratitude to Universiti Pendidikan Sultan Idris for the support/ facilities.

Fund
The authors would like to thank Research Management and Innovation Centre (RMIC) for providing a GPUF research grant (2021-0215-103-01) that enables the team to complete some part of the research objectives through this article.

Individual Assistant
NA

AUTHOR CONTRIBUTIONS

Annie Gorgey (Conceptualization; Methodology; Software; Analysis; Validation; Visualization; Review and Editing), Farah Adilla Azim (Literature Review; Methodology; Analysis - Original draft), Zul Hafiezy Zulkifly (Software and Analysis), Haslinah Hassim (Resources; Literature Review; Methodology).

DECLARATION OF ORIGINALITY

The authors declare no conflict of interest to report regarding this study conducted.

REFERENCES

[1] Gorgey, A. Numerical Methods in Scilab Programming. Tanjong Malim: UPSI Press; 2023.
[2] Atkinson, KE. An Introduction to Numerical Analysis (2nd ed.). Canada: John Wiley & Sons; 1991
[3] Reddy SS, Bijwe PR. An efficient optimal power flow using bisection method. Electrical Engineering. 2018;100(4):2217-2229.
[4] Burden, RL. and Faires, JD. Numerical Analysis. 10th Edition, Boston: Cengage Learning; 2015.
[5] Osman, Subhi, Rahman, Abdel, Mohammed, Adam, Ali, Abualez. Solution of non-linear equations using bisection method by new technical method. The Arab Journal of Educational and Psychological Sciences. 2022; 6(26): 261-274.
[6] Rosli NN, Shahari NA, Mohamad Azraei FA, Izaham SN. Comparative study of nonlinear root finding using improvised Secant methods. Malaysian Journal of Computing (MJoC). 2023;8(1):1332-1348.
[7] Chapra, SC., & Canale, RP. *Numerical Methods for Engineers* (8th ed.). McGraw-Hill Education. 2021.
[8] Ridders C. A new algorithm for computing a single root of a real continuous function. IEEE Transactions on circuits and systems. 2003;26(11):979-980.
[9] Rahman SK, Mohammed DA, Hussein BM, Salam BA, Mohammed KR, Faraj BM. An improved bracketing method for numerical solution of nonlinear equations based on ridders method. Matrix Science Mathematic. 2022;6(2):30-33.
[10] Noor MA, Khan WA, Hussain A. A new modified Halley method without second derivatives for nonlinear equation. Applied mathematics and computation. 2007;189(2):1268-1273.
[11] Galdino S. A family of regula falsi root-finding methods. In Proceedings of 2011 World Congress on Engineering and Technology (CET 2011) 2011 (vol. 1, pp. 514-517).
[12] Rao, SS. Mechanical Vibration. 5th Edition, Pearson Education, Inc., Upper Saddle River, 603-606: 2006.
[13] Norman SN. Control Systems Engineering. USA John Wiley & Sons: 2000.
[14] Nocedal, J, Wright, SJ. Numerical optimization. In Springer Series in Operations Research and Financial Engineering. (Springer Series in Operations Research and Financial Engineering). Springer Nature. 2006.
[15] Gorgey A, Zulkifly ZH, Hassim H, Azim FA. Comparison of some numerical methods for solving real-life nonlinear equations by using python programming. Journal of Mathematics and Computing Science. 2024;10(1):14-25.