eng

MDPI

Article

An Iterative Hybrid Algorithm for Roots of Non-Linear Equations

Chaman Lal Sabharwal

Computer Science Department, Missouri University of Science and Technology, Rolla, MO 65409, USA; chaman@mst.edu

Abstract: Finding the roots of non-linear and transcendental equations is an important problem in engineering sciences. In general, such problems do not have an analytic solution; the researchers resort to numerical techniques for exploring. We design and implement a three-way hybrid algorithm that is a blend of the Newton–Raphson algorithm and a two-way blended algorithm (blend of two methods, Bisection and False Position). The hybrid algorithm is a new single pass iterative approach. The method takes advantage of the best in three algorithms in each iteration to estimate an approximate value closer to the root. We show that the new algorithm outperforms the Bisection, Regula Falsi, Newton–Raphson, quadrature based, undetermined coefficients based, and decomposition-based algorithms. The new hybrid root finding algorithm is guaranteed to converge. The experimental results and empirical evidence show that the complexity of the hybrid algorithm is far less than that of other algorithms. Several functions cited in the literature are used as benchmarks to compare and confirm the simplicity, efficiency, and performance of the proposed method.

Keywords: bisection; false position; newton; order of convergence; predictor-corrector; quadrature

Citation: Sabharwal, C.L. An Iterative Hybrid Algorithm for Roots of Non-Linear Equations. *Eng* **2021**, *2*, 80–98. https://doi.org/10.3390/eng2010007

Academic Editor: Mohammad Reza Safaei

Received: 1 February 2021
Accepted: 25 February 2021
Published: 8 March 2021

1. Introduction

The construction of numerical solutions for non-linear equations is essential in many branches of science and engineering. Most of the time, non-linear problems do not have analytic solutions; the researchers resort to numerical methods. New efficient methods for solving nonlinear equations are evolving frequently, and are ubiquitously explored and exploited in applications. Their purpose is to improve the existing methods, such as classical Bisection, False Position, Newton–Raphson, and their variant methods for efficiency, simplicity, and approximation reliability in the solution.

There are various ways to approach a problem; some methods are based on only continuous functions while others take advantage of the differentiability of functions. The algorithms such as Bisection using midpoint, False Position using secant line intersection, and Newton–Raphson using tangent line intersection are ubiquitous. There are improvements of these algorithms for speedup; more elegant and efficient implementations are emerging. The variations of continuity-based Bisection and False Position methods are due to Dekker [1,2], Brent [3], Press [4], and several variants of False Position including the reverse quadratic interpolation (RQI) method. The variations of derivative based quadratic order Newton–Raphson method are 3rd, 4th, 5th, 6th order methods. From these algorithms, the researcher has to find a suitable algorithm that works best for every function [5,6]. For example, the bisection method for a simple equation $x - 2 = 0$, on interval [0, 4], will get the solution in one iteration. However, if the same algorithm is used on a different, smaller interval [0, 3], iterations will go on forever to get to 2; it will need some tolerance on error or on the number of iterations to terminate the algorithm.

For derivative based methods, several predictor–corrector solutions have been proposed for extending the Newton–Raphson method with the support of Midpoint (mean of endpoints of domain interval), Trapezoidal (mean of the function values at the endpoint of the domain), Simpsons (quadratic approximation of the function) quadrature

formula [7], undetermined coefficients [8], a *third order* Newton-type method to solve a system of nonlinear equations [9], Newton's method with accelerated convergence [10] using trapezoidal quadrature, *fourth order* method of undetermined coefficients [11], one derivative and two function evaluations [12], Newton's Method using *fifth order* quadrature formulas [13], using Midpoint, Trapezoidal, and Simpson quadrature *sixth* order rule [14]. Newton–Raphson and its variants use the derivative of the function; the derivative of the function is computed from integrations using quadrature-based methods. The new three-way hybrid algorithm presented here does not use any variations or quadrature or method of undetermined coefficient, and still, it competes with all these algorithms.

For these reasons, a two-way blended algorithm was designed and implemented that is a blend of the bisection algorithm and regula falsi algorithm. It does not take advantage of the differentiability of the function [15]. Most of the time, the equations involve differentiable functions. To take advantage of differentiability, we design a three-way hybrid algorithm that is a hybrid of three algorithms: Bisection, False Position, and Newton–Raphson. The hybrid algorithm is a new single pass iterative approach. The method does not use predictor–corrector technique, but predicts a better estimate in each iteration. The new algorithm is promising in outperforming the False Position algorithm and the Newton–Raphson algorithm. Table 1 is a listing of all the methods used for test cases. It is also confirmed [Tables 2–5] that it outperforms the quadrature based [7,13,14], undetermined coefficients methods [8], and decomposition based [16] algorithms in terms of number of function evaluations per iteration as well as overall number of iterations, computational order of convergence (COC), and efficiency index (EFF). This hybrid root finding algorithm performs fewer or at most that many iterations as these cited methods or functions. The bisection and regula falsi algorithms require only continuity and no derivatives. This algorithm is guaranteed to converge to a root. The hybrid algorithm utilizes the best of the three techniques. The theoretical and empirical evidence shows that the computational complexity of the hybrid algorithm is considerably less than that of the classical algorithms. Several functions cited in the literature are used to confirm the simplicity, efficiency, and performance of the proposed method. The resulting iteration counts are compared with the existing iterative methods in Table 2.

Even though the classical methods have been developed and used for decades, enhancements are made to improve the performance of these methods. A method may perform better than other methods on one dataset, and may produce inferior results on another dataset. We have seen an example just above that different methods have their own strengths/weaknesses. A dynamic new hybrid algorithm is presented here by taking advantage of the best in the Bisection, False Position, and Newton–Raphson methods to locate the roots independent of Dekker, Brent, RQI, and 3rd–6th order methods. Its computational efficiency is validated by comparing it with the existing methods via complexity analysis and empirical evidence. This new hybrid algorithm outperforms all the existing algorithms; see Section 4 for empirical outcomes validating the performance of the new algorithm.

This paper is organized as follows. Section 2 describes background methods to support the new algorithm. Section 3 is the new algorithm. Section 4 is experimental analysis using a multitude of examples used by researchers in the literature, results, and comparison with their findings. Section 5 is on the complexity of computations. Section 6 is the conclusion followed by references.

2. Background, Definitions

In order to review the literature, we briefly describe methods for (1) root approximation, (2) error calculation and error tolerance, and (3) algorithm termination criteria. There is no single optimal algorithm for root approximation. Normally we look for the solution in the worst-case scenario. The order of complexity does not tell the complete detailed outcome. The computational outcome may depend on implementation details, the domain, tolerance, and the function. No matter what, for comparison with different algorithms accomplishing the same task, we use the same function, same tolerance, same

termination criteria to justify the superiority of one algorithm over the other. When we are faced with competing choices, normally the simplest one is the accurate one. This is particularly true in this case. We provide a new algorithm that is simpler and outperforms all these methods. For background, we will briefly refer to two types of equations: (1) those requiring only continuous functions, and (2) those requiring differentiable functions along with the desired order of derivative in their formulations.

There are two types of problems: (1) continuity based with no derivative requirement, such as Bisection, False Position, and their extensions; and (2) derivative based, such as Newton–Raphson and its variations. For simulations, we use error Tolerance $\varepsilon = 0.0000001$, tolerance coupled with iterations termination criteria as $(|x_k - x_{k-1}| + |f(x_k)|) < \varepsilon$, and upper bound on iterations as 100: for function f:[a, b] $\rightarrow$ R, such that

(1) f(x) is continuous on the interval [a, b], where R is the set of all real numbers, and

(2) f(a) and f(b) are of opposite signs, i.e., $f(a) \cdot f(b) < 0$, then there exists a root $r \in [a, b]$ such that $f(r) = 0$, or

(2′) the function f(x) is differentiable with $g(x) = x - f(x)/f'(x)$ and $|g'(x) < 1$, then there exists a root $r \in [a, b]$ such that $f(r) = 0$.

Since the solution is obtained by iterative methods, the definition of convergence is as follows:

Definition (Convergence) [10,12,17]. *Let* x_n, *and* $\alpha \in R$, $n \geq 0$. *Then, the sequence* $\{x_n\}$ *is said to converge to* α *if*

$$\lim_{n \to \infty} |x_n - \alpha| = 1$$

Definition (**Order of Convergence**). *Let* x_n, *and* $\alpha \in R$, $n \geq 0$, *sequence* $\{x_n\}$ *converge to* α.

In addition, [12,18] if there exists a constant $C > 0$ ($C \neq 0$, $C \neq \infty$) and an integer $p \geq 1$ such that

$$\lim_{n \to \infty} \left(\frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^p} \right) = C$$

then the sequence $\{x_n\}$ is said to converge to α with convergence order p, and C is the asymptotic error constant [10,12,18,19].

Since $x_n = \alpha + e_n$, the error equation becomes

$$e_{n+1} = C\, e_n{}^p + O(e_n{}^{p+1})$$

If p = 1, 2, or 3, the convergence is called linear, quadratic, or cubic convergence, respectively.

These theoretical criteria do not take into consideration how much computation the function values perform in each step. The order of convergence p can be approximated by the Computational Order of Convergence (COC) that takes into account the combinatorial cost of the method.

Definition (Computational Order of Convergence). *Suppose three iterations* x_{n-1}, x_n, x_{n+1} *are closer to the root* α*; then, the order of convergence is approximated by [10,12,14]*

$$\mathrm{COC} = \lim_{n \to \infty} \left(\frac{\log \left| \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|} \right|}{\log \left| \frac{|x_n - \alpha|}{|x_{n-1} - \alpha|} \right|} \right)$$

Additionally, since α *is not known a priori, there are other ways to compute COC, namely, using four iterations [8] instead of three iterations [10,12,17].*

$$\mathrm{COC} = \lim_{n \to \infty} \left(\frac{\log \left| \frac{|x_{n+1} - x_n|}{|x_n - x_{n-1}|} \right|}{\log \left| \frac{|x_n - x_{n-1}|}{|x_{n-1} - x_{n-2}|} \right|} \right)$$

Definition (**Efficiency Index**). *If the power p is the order of convergence and q is the number of function evaluations per iteration, then* $p^{1/q}$ *is called the efficiency index and is denoted by EFF [10,12].*

2.1. Bisection Method

The Bisection method (1) is a binary approach for eliminating subintervals, (2) is virtually a binary search, and (3) converges slowly to the root. Without regard to the function, an upper bound on the root error is fixed at each iteration and it can be computed a priori. By specifying the tolerance on root error, the upper bound on the number of iterations is predetermined.

2.2. False Position (Regula Falsi) Method

The poor performance of the Bisection method led to the False Position method. It finds zeros of the function by linear interpolation. The False Position uses the location of the root for better selection. Unfortunately, this method does not perform as well as expected for all functions; see Figure 1 [15] for four types of concavity in the function. Iterations are terminated at ten to keep the plots clean. The figures show that the False Position method performs poorly as compared to the Bisection method.

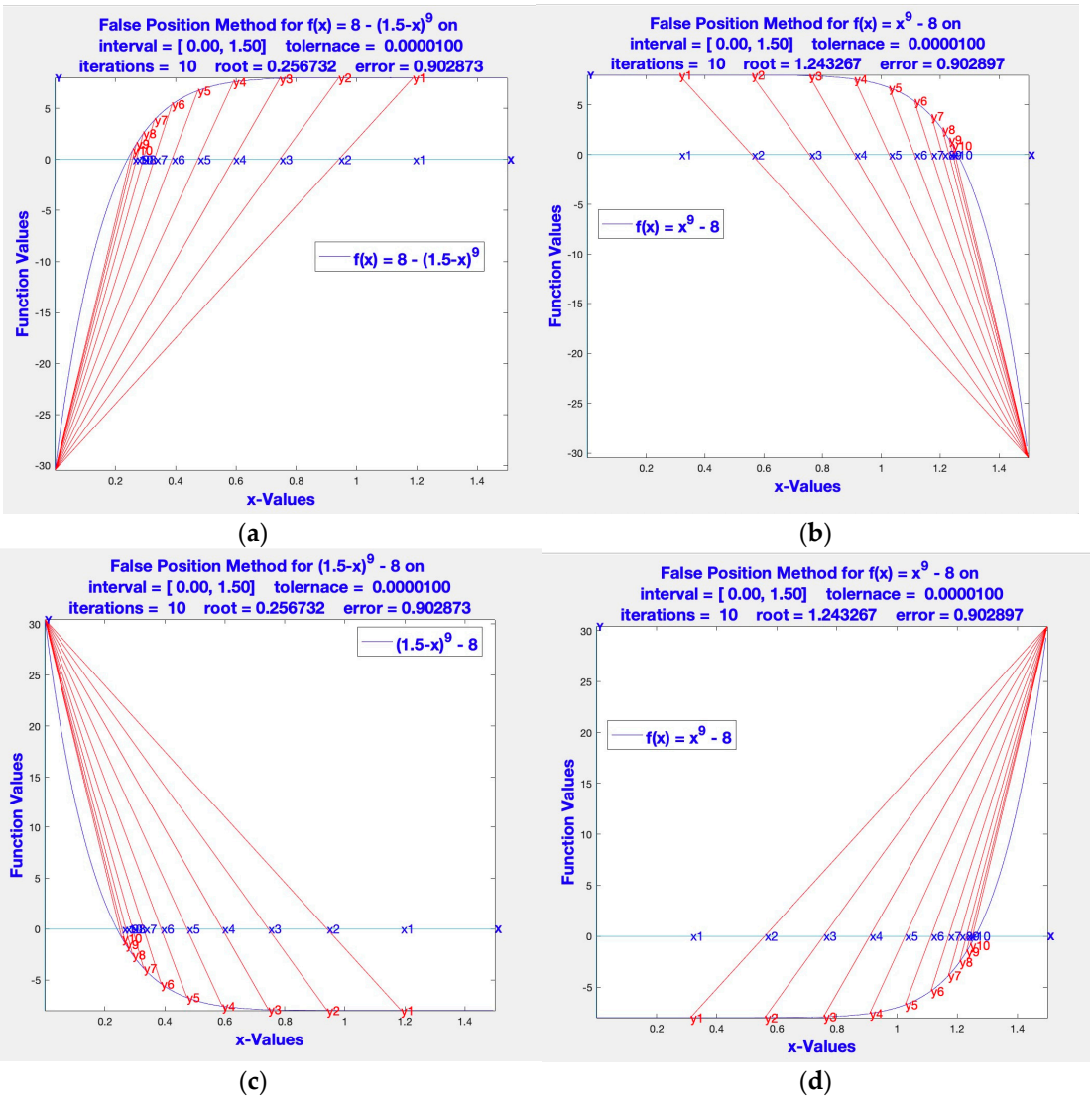

Figure 1. (**a**) Convex function concave up, left endpoint fixed. (**b**) Convex function concave up, right endpoint fixed. (**c**) Convex function concave down, left endpoint fixed. (**d**) Convex function concave down, right endpoint fixed.

2.3. Dekker's Method

Dekker algorithm [2] was designed to circumvent the shortcomings of the slow speed of the Bisection and the uncontrolled iteration count of the False Position method. It is a combination of two methods: Bisection and False Position. First of all, it assumes that f(x) is continuous on [a, b] and f(a)f(b) < 0. It maintains that $|f(b)| < |f(a)|$; otherwise, a and b are exchanged to ensure that b is the "best" estimate of the approximate root. The algorithm maintains that $\{b_k\}$ is the sequence of best estimates, and the root enclosing interval is always $[a_k, b_k]$ or $[b_k, a_k]$. Dekker's algorithm maintain three values: b is the current best approximation, c is the previous approximation b_{k-1}, and a is the contrapoint so that the root lies in the interval $[a_k, b_k] \cup [b_k, a_k]$. Both the secant point, s, and midpoint, m, are computed; b_k is s or m, whichever is closest to b_{k-1}. Though it is better than the False position method, nonetheless, it has some road blocks, handled by Brent.

2.4. Brent's Algorithm

Brent's algorithm [3] evolved from the failures of Dekker's algorithm; it is a step towards the improvement of Dekker's algorithm. It is a slight improvement at the cost of extra test computations. It is a combination of four methods, Bisection, False Position, Dekker, and reverse (a.k.a. inverse) quadratic interpolation (RQI), described next. Reverse quadratic interpolation is based on the method of Lagrange polynomials and it leads to extra calculations. Thus, it may take more iterations to circumvent pathological cases. It uses three estimates to derive the next estimate. This algorithm is more robust and more costly.

Detour to Reverse Quadratic Interpolation

Let a, b, and c be three distinct points, and f(a), f(b), and f(c) be corresponding distinct values of the function f(x); there is a unique quadratic polynomial through three points (a, f(a)), (b, f(b)), (c, f(c)). The Lagrange form of polynomial is most suitable for evaluation of the polynomial for values of x and inverse value of y. In fact, there is a direct formula for a reverse quadratic interpolating (RPI) polynomial that can be evaluated from values of y.

$$x = \frac{(y - f(b))(y - f(c))}{(f(a) - f(b))(f(a) - f(c))}a + \frac{(y - f(a))(y - f(c))}{(f(b) - f(a))(f(b) - f(c))}b + \frac{(y - f(a))(y - f(b))}{(f(c) - f(a))(f(c) - f(b))}c$$

or

$$x_{k+1} = \frac{af(b)f(c)}{(f(a) - f(b))\,(f(a) - f(c))} + \frac{bf(c)f(a)}{(f(b) - f(c))\,(f(b) - f(a))} + \frac{cf(a)f(b)}{(f(c) - f(a))\,(f(c) - f(b))}$$

The derivative based methods depend on the initial start point, x_0. The simplest such technique is the Newton–Raphson method, which is slower than its variations. These algorithms outperform the conventional Newton–Raphson method. The variations include decomposition based [16], quadrature based, and undetermined coefficients based [11,13,14]. These methods are quite complex and detailed. The reader may wish to refer to the full papers for details. For completeness, we describe the iteration formulas to show how these methods iterate to get to the root.

2.5. Newton-Raphson (1760)

Let f be differentiable on an open interval containing a, b. By Taylor series expansion of f,

$$f(b) = f(a) + (b - a)f'(a) + \frac{1}{2!}(b - a)^2 f''(a) + \ldots +$$

By retaining the first derivative term only, linear approximation of order one

$$f(b) \cong f(a) + (b - a)f'(a)$$

Assuming the value of b to be close to the root of f, further leads to

$$0 \cong f(a) + (b - a)f'(a)$$

$b = a - \frac{f(a)}{f'(a)}$ which is standard Newton–Raphson method.

Thus, for function f, Newton–Raphson [17] successive estimates for the solution are

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

with quadratic order of error, $O(\epsilon^2)$, where $\epsilon = |x_{n+1} - x_n|$.

This amounts to two function evaluations for each iteration. However, if we write

$$g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$$

then

$$x_{n+1} = g(x_n)$$

amounts to one function evaluation for each iteration. It is just a matter of how we count the number of function evaluations. It is of theoretical interest, but really, we have to consider the combinatorial cost of the function, too. In fact, we can write

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

That results in one function evaluation $f(x_n)$ in every iteration. We use this form in the algorithm.

2.6. Oghovese-John Method (2014)

The Oghovese–John sixth order method approximates the iteration estimates by using the average of Midpoint and Simpson quadrature, and it is shown to have approximation accuracy of order six.

The expression for Oghovese–John's estimates [7] is as follows.

Let

$$u_0 = x_0 - \frac{f(x_0)}{f'(x_0)} v_0 = u_0 - \frac{f(u_0)}{f'(u_0)}$$

Then, for $n \geq 0$

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)} \tag{2}$$

$$v_{n+1} = v_n - \frac{12\, f(v_n)}{f'((v_n) + 10\, f'(\frac{(v_n + u_{n+1})}{2}) + f'(u_{n+1})} \tag{3}$$

(Based on the average of Midpoint and Simpsons 3/8 rule.)

Then, the iterates, x_{n+1}, are defined in terms of v_n instead of x_n

$$x_{n+1} = v_n - \frac{f(v_n)}{f'(v_n)} \text{ for } n \geq 0 \tag{4}$$

instead of Newton–Raphson formula $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

Convergence and stopping criteria are specified with error tolerance and upper bound on iterations for termination. In fact, the algorithm terminates considerably much earlier than it reaches any termination condition in the algorithm.

The derivation of the Oghovese–Emunefe [7] method uses the average of Midpoint and Simpson quadrature. We briefly describe its derivation in Appendix A.

The following variations are very interesting and challenging. It is a historical perspective [18–24] of development of innovations in the improvement of the Newton–Raphson method and its variants for solution to non-linear equations. We will compare the hybrid algorithm with these. First, we describe their iteration recurrence relations without derivations and defer their derivations to references for simplicity.

2.7. Grau-Diaz-Barero (2006)

Grau et al. [18] introduced a new method by correcting Ostrowski's method [12]:

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

$$x_{n+1} = y_n - \frac{f(y_n)}{f'(x_n)} \frac{2f(x_n) - f(y_n)}{2f(x_n) - 5\, f(y_n)} \tag{6}$$

The method uses one derivative and two function evaluations.
The method has a *sixth order* convergence with the following iterations:

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)} \tag{7}$$

$$z_n = y_n - \frac{f(y_n)}{f'(x_n)} \frac{f(x_n)}{f(x_n) - 2\, f(y_n)} \tag{8}$$

$$x_{n+1} = z_n - \frac{f(z_n)}{f'(x_n)} \frac{f(x_n)}{f(x_n) - 2f(y_n)} \tag{9}$$

2.8. Sharma-Guha (2007)

Sharma and Guha [22] modified and parameterized Ostrowski's method [12] having four function evaluations and a *sixth order* convergence. Their formula is given as follows:

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)} \tag{10}$$

$$z_n = y_n - \frac{f(y_n)}{f'(x_n)} \frac{f(x_n)}{f(x_n) - 2\, f(y_n)} \tag{11}$$

$$x_{n+1} = z_n - \frac{f(z_n)}{f'(x_n)} \frac{f(x_n) + af(y_n)}{f(x_n) + bf(y_n)} \tag{12}$$

where a and b are problem dependent parameters, with b = a − 2.

2.9. Khattri-Abbasbandy (2011)

Khattri-Abbasbandy [11] introduced an iterative method having a fourth order convergence using three function evaluations per iteration with the following formula:

$$y_n = x_n - \frac{2}{3} \frac{f(x_n)}{f'(x_n)} \tag{13}$$

$$x_{n+1} = y_n - [1 + \frac{21}{8} \frac{f'(y_n)}{f'(x_n)} + \frac{-9}{2} (\frac{f'(y_n)}{f'(x_n)})^2 + \frac{15}{8} (\frac{f'(y_n)}{f'(x_n)})^3] \frac{f(x_n)}{f'(x_n)} \tag{14}$$

2.10. Fang-Chen-Tian-Sun-Chen (2011)

Fang et al. [23] modified Newton's method with five evaluation functions and produced a sixth order convergence method having the following iterations:

$$y_n = x_n - \frac{f(x_n)}{a_n\, f(x_n) + f'(x_n)} \tag{15}$$

$$z_n = y_n - \frac{f(y_n)}{b_n f(y_n) + f'(y_n)} \tag{16}$$

$$x_{n+1} z_n - \frac{f(z_n)}{a_n f(z_n) + f'(z_n)} \tag{17}$$

where is a_n, b_n, c_n are real numbers chosen in such a way that $0 \leq |a_n|, |b_n|, |c_n| \leq 1$, and $\text{sign}(a_n f(x_n)) = \text{sign}(f'(x_n))$, $\text{sign}(b_n f(y_n)) = \text{sign}(f'(y_n))$,

$$\text{sign}(c_n f(Z_n)) = \text{sign}(f'(z_n)),$$

where $n = 1,2, \ldots$, and sign(x) is a sign function.

2.11. Jayakumar (2013)

Jayakumar generalized Simpson–Newton's [14] method for solving nonlinear equations. His algorithm has *third order* convergence and four function evaluations per iteration.

Let a be a root of the function (1) and suppose that x_{n-1}, x_n, x_{n+1} are three successive iterations closer to the root a. This is a third order convergence formulation.

Recall the Simpson 1/3 iteration formula

$$x_{n+1} = x_n - \frac{6 f(x_n)}{f'(x_n) + 4 f'(\frac{x_n + y_n}{2}) + f'(y_n)} \tag{18}$$

There is no end in sight from the extensions; the arithmetic mean $\frac{f'(x_n) + f'(y_n)}{2}$ is implicit Equation (21). Harmonic–Simpson–Newton's method (HSN): In Equation (21), using harmonic mean instead of arithmetic mean $\frac{f'(x_n) + f'(y_n)}{2}$, the Harmonic–Simpson–Newton's method becomes

$$x_{n+1} = x_n - \frac{3 f(x_n)}{\frac{2 f'(x_n) f'(y)}{f'(x_n) + f'(y_n)} + 2 f'(\frac{x_n + y_n}{2})} \tag{19}$$

2.12. Nora-Imran-Syamsudhuha (2018)

This is a very interesting and complex analysis paper [8], the order of convergence is still six. In this article, the authors present a combination of the Khattri and Abbasbandy [11] method with Newton's method using the principle of undetermined coefficients method.

Furthermore, using the *fourth order* method derived by Khattri and Abbasbandy, they propose the following iterative method.

$$y_n = x_n - \frac{2}{3} \frac{f(x_n)}{f'(x_n)} \tag{20}$$

$$z_n = y_n - [1 + \frac{21}{8} \frac{f'(y_n)}{f'(x_n)} + \frac{-9}{2} (\frac{f'(y_n)}{f'(x_n)})^2 + \frac{15}{8} (\frac{f'(y_n)}{f'(x_n)})^3] \frac{f(x_n)}{f'(x_n)} \tag{21}$$

$$x_{n+1} = z_n - [\frac{ab(a-b)f(z_n)}{m f'(x_n) - a^3 f'(y_n) + 2b(2a-b)(f(z_n) - f(x_n))'}] \tag{22}$$

with $m = a(b^2 - 3ab + a^2)$, $a = z_n - x_n$, $b = y_n - x_n$.

2.13. Weerakon et al. (2000)

Weerkoon–Fernando [10] used trapezoidal quadrature to achieve third order convergence with the following iteration forms

The trapezoid rule is

$$(b - a) \{ f'(a) + f'(b) \}/2 \cong -f(a)$$

$$b \cong a - \frac{2\,f(a)}{\{f'(a) + f'(b)\}} \tag{23}$$

The Weerakoon–Fernando formula becomes

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)} \tag{24}$$

$$x_{n+1} = x_n - \frac{2\,f(x_n)}{f'(x_n) + f'(y_n)} \text{ or}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{(f'(x_n) + f'(y_n))/2} \tag{25}$$

This is also called the *Arithmetic mean formula*. If arithmetic mean is replaced with Harmonic mean, another variation is called the *Harmonic mean formula*

$$x_{n+1} = x_n - \frac{f(x_n)}{\frac{2}{(\frac{1}{f'(x_n)} + \frac{1}{f'(y_n)})}} \tag{26}$$

$$x_{n+1} = x_n - \frac{\frac{1}{2}f(x_n)(f'(x_n) + f'(y_n))}{f'(x_n)\,f'(y_n)} \tag{27}$$

2.14. Edmond Halley (1995)

Halley [24] improved Newton's method. Halley's method (1995) requires that the function be C^3, three times continuously differentiable. The root iterations have cubic convergence. The function f(x) is expanded to approximate the quadratic in two ways and cancelling the second degree term to arrive at the linear formula

$$0 = f(x_{n+1}) = f(x_n) + (x_{n+1} - x_n)f'(x_n) + \frac{(x_{n+1} - x_n)^2}{2} f''(x_n) + O((x_{n+1} - x_n)^3) \tag{28}$$

$$0 = f(x_{n+1}) = f(x_n) + (x_{n+1} - x_n)f'(x_n) + O((x_{n+1} - x_n)^2) \tag{29}$$

Multiply (28) by $(x_{n+1} - x_n)f''(x_n)$, (29) by $2f'(x_n)$,
and subtract the resulting (28) from (29); we get

$$x_{n+1} = x_n - \frac{2\,f(x_n)\,f'(x_n)}{2\,f'(x_n)^2 - f(x_n)\,f''(x_n)} \tag{30}$$

Halley's algorithm has convergence of order three. This completes our discussion of the derivative based formulas.

In summary, the methods defined in Sections 2.5–2.14 are several variations of the Newton–Raphson method. Table 1 is a descriptions of the symbols used in the Tables 2–5, to identify the methods: MN_R stands for Newton–Raphson method, MOJmis stands for Oghovese–John method; (it uses Midpoint_Simpson1/3 method), MGDB stands for Grau-Diaz–Barero method, MSG stands for Sharma–Guha method, MKA stands for Khattri–Abbasbandy method (which uses the method of undetermined coefficients), FCTSC stands for Fang–Chen–Tian–Sun–Chen method, MJKms, MJKhs stand for JayaKumar method, two versions (he uses mean and Simpson1/3 rule; also, he uses harmonic mean with Simpson 1/3 rule), MNIS stands for Nora–Imran–Syamsudhuha method (it extends Khattri and Abbasbandy method), NWFhm3 stands for Weerakon et al. method using harmonic mean, NHAL stands for Edmond Halley method using second derivatives, Hybridn stands for hybrid method using Bisection, False Position, and Newton's methods. Hybrid method is the three way method described next in Section 3.

Table 1. Methods used in Tables 2–5 with Efficiency Index (EFF).

Method	Year	Name in Tables 2–5	EFF
Newton–Rapson	1760	MN_R	1.4142
Edmond Halley	1995	NHAL	1.4422
Weerkoon–Fernando	2000	NWFhm3	1.4310
Grau–Diaz–Barero	2006	MGDB	1.5651
Sharma–Guha	2007	MSG	1.5651
Fang–Chen–Tian–Sun–Chen	2011	FCTSC	1.3480
Khattri–Abbasbandy	2011	MKA	1.4860
Jayakumar	2013	MJKs	1.3161
Jayakumar	2013	MJKhs	1.3161
Oghovese–John	2014	MOJmis	1.2599
Nora–Imran–Syamsudhuha	2018	MNIS	1.5651
Hybrid Method	2021	Hybridn	1.5874

3. Three Way Hybrid Algorithm

The three way algorithm is an extension of the two way algorithm [15] used for continuous functions. The three way algorithm takes advantage of the differentiability of the function, also. First, the bisection and false position methods compete for a more accurate approximate root; then, the algorithm computes the smaller enclosing interval for it. At the next step, the better of the approximate root and Newton root are processed to get the better of the three methods. The algorithm is as follows and the Matlab code for the hybrid algorithm in given in Appendix B.

Hybrid Algorithm

Output: root r, k-number of iterations used, bracketing interval $[a_{k+1}, b_{k+1}]$
//initialize
$k = 0$; $a_1 = a$, $b_1 = b$
Initialize bounded interval for bisection and false position
fa is false position a, ba is bisection method a
$fa_{k+1} = ba_{k+1} = a_1$; $fb_{k+1} = bb_{k+1} = b_1$
$n_1 = a_1 - f(a_1)/\ f'(a_1)$;
repeat
$fa_{k+1} = ba_{k+1} = a_k$; $fb_{k+1} = bb_{k+1} = b_k$
$n_{k+1} = n_k - f(n_k)/f'(n_k)$
/compute the midpoint
$m = \frac{a_k + b_k}{2}$, and $\in_m = |f(m)|$
compute the False Position secant line point,
$s = a_k - \frac{f(a_k)\ (b_k - a_k)}{f(b_k) - f(a_k)}$ and $\in_f = |f(s)|$
$r = s$
$\in_a = \in_f$
if $|f(m)| < |f(s)|$,
f(m) is closer to zero, Bisection method determines bracketing interval $[ba_{k+1}, bb_{k+1}]$
$r = m$, $\in_m = f(m)$
$\in_a = \in_m$
if $f(a_k) \cdot f(r) > 0$,
$ba_{k+1} = r$; $bb_{k+1} = b_k$;
else
$ba_{k+1} = a_k$; $bb_{k+1} = r$;
else
f(s) is closer to zero, False Position method determines bracketing interval $[fa_{k+1}, fb_{k+1}]$
$r = s$, $\in_f = f(s)$

$\in_a = \in_f$
if $f(a_k) \cdot f(r) > 0$,
$fa_{k+1} = r$; $fb_{k+1} = b_k$;
else
$fa_{k+1} = a_k$; $fb_{k+1} = r$;
endif
endif
Since the root is bracketed by both $[ba_{k+1}, bb_{k+1}]$ and $[fa_{k+1}, fb_{k+1}]$, define
$[a_{k+1}, b_{k+1}] = [ba_{k+1}, bb_{k+1}] \cap [fa_{k+1}, fb_{k+1}]$
$a_{k+1} = \max(ba_{k+1}, fa_{k+1})$;
$b_{k+1} = \min(bb_{k+1}, fb_{k+1})$;
//if f is differentiable
//use n_{k+1} if $f(n_{k+1}) < \min(f(a_{k+1}), f(b_{k+1}))$
//replace $a_{k+1.}$ or b_{k+1} by n_{k+1} resulting in further smaller interval, with new root $r = n_{k+1}$
outcome: iteration complexity, root, and error of approximation
iterationCount = k
$r_{k+1} = r$
$k = k + 1$
error = $\in_a = |f(r_k)| + |b_k - a_k|$
until $\in_a < \in$ or k > maxIterations

In Section 5, for comparisons, this algorithm is labelled hybridn and its implementation Matlab code is given in the Appendix B.

4. Convergence of Hybrid Algorithm

The new hybrid algorithm is an improvement over the Newton–Raphson algorithm and a blend of two algorithms: Bisection algorithm and False Position algorithm, and is independent of any other algorithm. The new algorithm differs from all the previous algorithms by tracking the best root approximation in addition to the best bracketing interval. The number of iterations to find a root depends on the criteria used to determine the root accuracy. The complexity of the new algorithm is better that the bisection. In other words, it uses fewer iterations than the bisection algorithm by retaining the root bracketed, whereas other algorithms use more iterations than the Bisection method. If f(x) is used, then complexity depends on the function as well as the method, because we want to ensure that the function value at the estimate is tolerable.

If relative error is used, it does not work for every function because it gets stuck for the case where relative error is constant. Most of the time, absolute error, $\in_s$, is used as the stopping criteria. For the Bisection method, on interval [a, b], the upper bound $n_b(\in)$ on the number of iterations can be found from $\frac{b-a}{2^n} < \in_s$ and is $\lg((b - a)/\in_s)$. Since $e_{n+1} = 1/2\ e_n$, it has linear convergence. For the False Position method, it depends on the location of the root near the endpoint of the bracketing interval and the convexity of the function. Thus, the bound $n_f(\in_s)$ for the number of iterations for the False Position method cannot be predetermined, it can be less, $n_f(\in_s) < n_b(\in) = \lg((b - a)/\in_s)$, or can be greater, $n_f(\in_s) > n_b(\in_s) = \lg((b - a)/\in_s)$. The number of iterations, $n(\in_s)$, in the hybrid algorithm is less than $\min(n_f(\in_s), n_b(\in_s))$. The introduction of the Newton–Raphson in the Hybrid further reduces the complexity of computations, resulting in fewer iterations. The Newton–Raphson algorithm of the quadratic order of convergence is given in Section 2.5: The convergence analysis of Newton is trivial; however, for completeness it is as follows.

Proposition. *α is a simple real root of a sufficiently differentiable function f on an open interval (a, b) of real line. If, then, the initial estimate x_0 is sufficiently close to α, then the Newton-Raphson as defined in Section 2.5 has second order convergence.*

Proof. It is trivial; for completeness, we derive the formula formally.

If α is a root, x_n is the approximation, then $f(\alpha) = 0$, $x_n = \alpha + e_n$ with error e_n. Let $c_k = \frac{1}{k!}\frac{f^{(k)}(\alpha)}{f'(\alpha)}$, then by Taylor series

$$\begin{aligned} f(x_n) &= f'(\alpha)[e_n + c_2\, {e_n}^2 + O({e_n}^3)] \\ f'(x_n) &= f'(\alpha)[1 + 2c_2\, e_n + 3C_c\, {e_n}^2 + O({e_n}^3)] \end{aligned}$$

Since α is a simple root $f'(\alpha) \neq 0$.

By dividing we get,

$$\frac{f(x_n)}{f'(x_n)} = e_n - c_2\, {e_n}^2 + O({e_n}^3)$$

From Newton–Raphson recurrence

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}\, x_n - e_n + O({e_n}^2) = \alpha + O({e_n}^2).$$

Hence, the order of convergence is quadratic. □

Since the hybrid algorithm is a combination of three algorithms, no derivatives are involved in the Bisection and False Position, and Newton's method can also use the secant approach to avoid derivatives in computations. The convergence order of the three methods is one, one, and two; they may be combined to get the composite order of convergence. The order of convergence that takes into account the combinatorial cost is computed. It competes with other sixth order methods. Thus, the COC indicates that the order of convergence is closer to sixth order, Table 5. Even if the order of convergence is taken as five or four, the efficiency index competes with them. We safely assume the fourth order for the hybrid algorithm, which is supported by Tables 2–5. We have convergence order four and three function evaluations as shown in (Section 2.5). The efficiency Index is computed with convergence order four and three function evaluations per iteration. The new algorithm complexity is far simpler than the higher order algorithms. This is validated with the experimental computations based on the number of computational iterations, see Tables 2–5.

This algorithm guarantees the successful resolution of the roots of non-linear equations. Other variants of Newton–Raphson converge with 3rd–6th order; they are not guaranteed to converge without additional constraints on the functions and the initial guess is close to the root. It differs from all the previous algorithms by tracking the best bracketing interval in addition to the best root approximation. Instead of brute force application of the Bisection or False Position or Newton–Raphson methods solely, we select the most relevant method and use its value at each step to construct the approximate root and bracketing interval. Thus, we design a new hybrid algorithm that performs better than the Bisection method, False Position method, and Newton–Raphson methods. Since an equation can have several roots, the user can appropriately isolate an interval enclosing a single root. Otherwise, different algorithms will end up reaching different roots, making it difficult to compart the algorithms. The new algorithm also performs better than the other variants cited above for continuous functions as long as the $f(a)\cdot f(b) < 0$ is determined at the end points of the defining interval. At each iteration, the root estimate is computed from both midpoint, secant point, tangent point if differentiable, and the better of the three is selected for the next approximation; in addition, the common enclosing interval is updated to the new smaller interval. This eliminates the unnecessary iterations in either method. There is no requirement on differentiability but derivative used only if available as required by Newton–Raphson and its variants. This is a simple, reliable, and fast algorithm.

5. Empirical Results of Simulations

The new hybrid algorithm has been tested to ensure that it performs better than other existing methods by optimizing the number of iterations required for approximations, the computation order of convergence, and the efficiency index for the test cases. There are various types of functions: polynomial, trigonometric, exponential, rational, logarithmic, and a heterogeneous combination of these.

The methods defined in Sections 2.5–2.14 are several variations of the Newton–Raphson method. The symbols used to identify the methods are described in Table 1. For comparison with the hybrid algorithm, the tables have several parameters: function, initial value, order of convergence, number of function values per iteration (nofe), Number of iteration (NIter), overall number of function evaluations (NOFE), computational order of convergence (COC), efficiency index (EFF), root, error, between the last two iterations and function value. The hybrid algorithm is labelled Hybridn where [a,b] is the interval of definition and a is used as the initial start value for all other algorithms. All tables show that the hybrid algorithm performs better with respect to number of iterations, number of function evaluations, efficiency index, and computational order of convergence. The hybrid algorithm computes the root with fewer than or equal to the number of iterations of other algorithms as evidenced in the tables.

We have compared the hybrid algorithm with most of the functions used in the literature. These methods were extensively tested on all the functions in Table 2.

Since it is multi-dimensional data, two dimensional table includes a specific feature related to all other parameters. The 22 functions have been tested on all 10 parameters and all 12 methods. Table 2 compares one parameter NOFE for all algorithms on all functions. Instead of 22 displays tables, the next three tables compare all parameters using a different function for NOFE, COC, and EFF. Table 3 compares NOFE on all algorithms and on all parameters with one function. Table 4 compares COC on all algorithms and on all parameters with another function. Table 5 compares EFF on all algorithms and on all parameters with another different function.

The tables show that the hybrid algorithm performs better than other algorithms with respect to NOFE, COC, and EFF and the hybrid algorithm comes ahead. In Table 2, for each function, a different interval was used depending on the function definition. For example, in Table 3, the function is $\sin(x) - x^3$, the initial interval value is [0.5, 1]; in Table 4, the function is $0.7x^5 - 8x^4 + 44x^3 - 90x^2 + 82x - 25$, the initial interval value is [0, 1], and in Table 5, the function is $x^3 + \log(x)$; the initial interval value is [0.1, 2].

Table 2. One Parameter. Comparison of overall number of function evaluations (NOFE) in hybrid algorithm and all other algorithms on all functions for the number of function evaluations required for the solution.

Function	$x^3 - x^2 - x - 1$	$\exp(x) + x - 6$	$x^3 + \log(x)$	$\sin(x) - x/2$	$(x-1)^3 - 1$	$x^3 + 4x^2 - 10$	$(x-2)^{23} - 1$	$\sin(x)^2 - x^2 + 1$	$8 - x + \log2(x)$	$1.0/(x-3) - 6$	$x^9 - 8$	$x^{0.5} - 1$	$0.7x^5 - 8x^4 + 4\,4x^3 - 90x^2 + 82x - 25$	$5x^3 - 5x^2 + 6x - 2$	$0.5x^3 - 4x^2 + 5.5\,x - 1$	$5x^3 - 5x^2 + 6x - 2$	$-0.6x^2 + 2.4x + 5.5$	$x^8 - 1$	$\sin(x) - x^3$	$8\exp(-x)\sin(x) - 1$	$\sin(x) + x$	$(0.8 - 0.3*x)/x$
Method																						
MN_R	34	14	10	10	6	8	22	16	14	18	42	14	12	8	12	8	6	52	14	6	6	10
MNIS	40	44	60	76	28	28	84	76	64	60	180	220	44	60	76	76	68	84	44	56	28	60
MKA	27	27	39	51	15	18	57	51	42	42	153	114	42	39	63	63	45	51	21	39	21	33
MSG	28	12	39	8	8	8	64	28	12	12	36	12	12	8	32	8	8	248	16	8	12	12
FCTSC	600	12	12	12	6	6	18	12	36	18	60	18	12	12	18	12	12	18	12	6	18	18
MGDB	20	8	32	8	8	8	16	12	12	12	36	16	12	8	32	8	8	36	12	4	12	12
MOJmis	81	15	9	9	6	6	39	21	21	18	42	12	12	6	9	6	6	3	15	6	12	12
MWFhm3	50	10	10	10	5	10	15	15	20	15	40	15	10	10	10	10	5	40	10	5	10	10
MJKs	36	20	12	12	8	8	72	28	28	24	56	20	16	12	20	12	8	4	24	8	8	16
MJKhs	36	16	12	12	8	8	28	24	28	20	52	16	16	8	12	8	8	4	12	8	12	12
NHAL	27	12	27	12	9	9	15	18	27	9	198	45	12	12	18	12	15	18	18	9	30	24
Hybridn	15	6	6	6	3	6	12	9	9	9	24	9	6	6	6	6	3	3	9	3	3	9

Table 3. All Parameters highlighting NOFE. Comparison of hybrid and all algorithms on all parameters used for the solution for a function.

Summary for Comparison of Methods for									
Function $\sin(x) - x^3$, Intial value 0.500									
Max Iterations = 100			Error tolerance = 0.0000001000						
Method	Order	nofe	NIters	NOFE	COC	EFF	Root	$\lvert x_n - x_{n-1} \rvert$	Function Value
MN_R	2	2	7	14	2.01	1.4142136	−0.928626	6.547×10^{-5}	0.000000014
MNIS	6	4	11	44	1	1.5650846	−0.928626	1.04×10^{-7}	−0.000000085
MKA	4	3	7	21	1	1.5874011	0.9286263	9.7×10^{-8}	0.000000077
MSG	6	4	4	16	4.29	1.5650846	−0.928626	1.13×10^{-6}	0
FCTSC	6	6	2	12	4.29	1.3480062	0	0	0
MGDB	6	4	3	12	4.29	1.5650846	−0.928626	2.56×10^{-7}	0
MOJmis	2	3	5	15	3.93	1.2599211	0.9286263	0.000183	0
MWFhm3	6	5	2	10	3.93	1.4309691	−0.928626	0	0
MJKs	3	4	6	24	3.42	1.316074	0.9286263	1.302×10^{-5}	0
MJKhs	3	4	3	12	4.79	1.316074	0.9286263	0.0022525	0.000000039
NHAL	3	3	6	18	3	1.4422496	0	0	0
Hybridn	4	3	3	9	4.83	1.5874011	−0.928626	6.547×10^{-5}	0

Table 4. All Parameters highlighting computational order of convergence **(COC).** Comparison of hybrid and all algorithms on all parameters used for the solution for another function.

Summary for Comparison of Methods for									
Function $0.7*x^5 - 8*x^4 + 44*x^3 - 90*x^2 + 82*x - 25$, Initial Value 00									
Max Iterations = 100			Error tolerance = 0.0000001000						
Method	Order	nofe	NIters	NOFE	COC	EFF	Root	$\lvert x_n - x_{n-1} \rvert$	Function Value
MN_R	2	2	6	12	2.01	1.414214	0.579409	1×10^{-7}	0
MNIS	6	4	11	44	1	1.565085	0.579409	1.5×10^{-8}	-9.9×10^{-8}
MKA	4	3	13	39	1	1.587401	0.579409	1.2×10^{-8}	-7.9×10^{-8}
MSG	6	4	3	12	5.79	1.565085	0.579409	9.5×10^{-8}	0
FCTSC	6	6	2	12	5.79	1.348006	0.579409	7.9×10^{-8}	-7.9×10^{-8}
MGDB	6	4	3	12	5.79	1.565085	0.579409	1.1×10^{-8}	0
MOJmis	2	3	4	12	2.94	1.259921	0.579409	7.83×10^{-7}	0
MWFhm3	6	5	2	10	2.94	1.430969	0.579409	0	0
MJKs	3	4	4	16	2.93	1.316074	0.579409	1.16×10^{-6}	0
MJKhs	3	4	4	16	2.96	1.316074	0.579409	1.02×10^{-7}	0
NHAL	3	3	4	12	3.01	1.44225	0.579409	1.6×10^{-8}	0
Hybridn	4	3	2	6	6.76	1.587401	0.579409	0	0

Table 5. All Parameters highlighting efficiency index **(EFF).** Comparison of hybrid and all algorithms on all parameters used for the solution for another different function.

Summary for Comparison of Methods for									
Function $x^3 + \log(x)$, Initial value 0.100									
Max Iterations = 100			Error tolerance = 0.0000001000						
Method	Order	nofe	NIters	NOFE	COC	EFF	Root	$\lvert x_n - x_{n-1} \rvert$	FunctionValue
MN_R	2	2	5	10	1.94	1.4142	0.704709	3.57×10^{-7}	0
MNIS	6	4	15	60	1	1.5651	0.036264	4.7×10^{-8}	2.9×10^{-8}
MKA	4	3	13	39	1	1.5874	0.704709	6×10^{-8}	−0.00000007
MSG	6	4	7	28	6.62	1.5651	0.036264	3.53×10^{-7}	0
FCTSC	6	6	2	12	6.62	1.3480	0.704709	0	0
MGDB	6	4	8	32	3.22	1.5651	0.704709	0.059996	5×10^{-9}
MOJmis	2	3	3	9	4.59	1.2599	0.704709	0.00025	0
MWFhm3	6	5	2	10	5.79	1.4310	0.704709	0	0
MJKs	3	4	3	12	6.34	1.3161	0.704709	0.000155	0
MJKhs	3	4	3	12	5.71	1.3161	0.704709	8.56×10^{-5}	0
NHAL	3	3	9	27	3.04	1.4422	0.704709	0	0
Hybridn	4	3	2	6	6.87	1.5874	0.704709	0	0

The results show that the new algorithm competes with the algorithms presented in Sections 2.5–2.14. This algorithm has the reliability of the Bisection method. This algorithm does not get stranded as do many of the cited algorithms into complexity of computations, resulting in slow speed. It has the speed of the False Position and Newton–Raphson [25] methods, which works all the time by ensuring the root bounds. The efficiency index and computational order of convergence also confirm the superiority of this algorithm. This is a simple, reliable, and fast algorithm.

This paper deals with single scalar variable no-linear equations. There are similar multi-dimensional vector valued systems of non-linear equations [26]. It will be interesting to explore its application in the future. There is an interesting application of numerical solutions in differential equations [27]. Exploring differential equations is not the purpose of this paper; we will investigate this phenomenon in future work.

6. Conclusions

This paper implements a new algorithm, a hybrid combination of Bisection method, Regula Falsi method, and Newton–Raphson method. We implemented the algorithm in Matlab R2018B 64 bit (maci64) on MacBook Pro MacOS Mojave2.2GHz intel Core i716 GB2400MHz DDR4 Radeon Pro555X 4GB. The implementation experiments indicate that hybrid algorithm outperforms three algorithms all the time. It is also determined that the new algorithm competes with the Newton–Raphson method and its variants both in computational order of convergence and efficiency index. The experiments on numerous datasets used in the literature justify that the new algorithm is effective. Thus, the paper contributes a superior new algorithm to the repertoire of classical algorithms.

Funding: It was not funded by any organization.

Institutional Review Board Statement: No IRBS was required to publish this academic research.

Informed Consent Statement: Not applicable.

Data Availability Statement: No special data was used. Everything is in public domain.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The derivation of the Oghovese–Emunefe [7] method uses the average of Midpoint and Simpson quadrature. We briefly describe its derivation and will defer the details of derivation for other methods in Sections 2.6–2.14 to the references. The reader may refer to the full paper for details of the order of convergence. By fundamental theorem of integration,

$$I = \int_a^b f'(t)\,dt = f(b) - f(a) \tag{A1}$$

For approximating b as a root, that is, f(b) ≅0,

$$\int_a^b f'(t)\,dt \cong -f(a) \tag{A2}$$

Using integration by standard quadrature forms,

By Midpoint rule

$$(b-a)f'\left(\frac{a+b}{2}\right) \cong -f(a) \tag{A3}$$

$$b \cong a - \frac{f(a)}{f'\left(\frac{a+b}{2}\right)} \tag{A4}$$

By Simpson quadratic approximation 3/8 rule

$$\left(\frac{b-a}{6}\right)\{f'(a) + 4f'\left(\frac{a+b}{2}\right) + f'(b)\} \cong -f(a) \tag{A5}$$

$$(b-a) \cong -\frac{6f(a)}{f'(a) + 4f'\left(\frac{a+b}{2}\right) + f'(b)}$$

$$b \cong a - \frac{6f(a)}{f'(a) + 4f'\left(\frac{a+b}{2}\right) + f'(b)} \tag{A6}$$

By applying the weighted average approximation rule on (3) and (5), using Midpoint and Simpson 3/8 rule for M and S,

$$-\ f(a)\ \cong \int_a^b f'(t)\,dt\ =\ (1\ -\ p)M\ +\ pS \text{ with } 0\ \leq\ p\ \leq\ 1$$

With p = 1/2

$$-\ f(a) \cong 1/2\ (b-a)\ f'(\frac{a+b}{2})\ +\ 1/2\ (\frac{b\ -\ a}{6})\{f'(a)\ +\ 4\ f'(\frac{a\ +\ b}{2}) + f'(b)\} \tag{A7}$$

Becomes

$$b\ \cong\ a\ -\ \frac{12\ f(a)}{f'(a) + 10\ f'(\frac{a+b}{2}) + f'(b)} \tag{A8}$$

Now the method proceeds to predict-correct the estimate as follows.

Using the predicted value $\overline{b} \cong a - \frac{f(a)}{f'(a)}$ in the right above, to correct it,

Ogbereyivwe et al. have first approximation computed as follows

$$\overline{b} \cong a\ -\ \frac{f(a)}{f'(a)} \qquad (\text{predicted}) \tag{A9}$$

$$\overline{\overline{b}} \cong\ a\ -\ \frac{12\ f(a)}{f'(a)\ +\ 10\ f'(\frac{a+\overline{b}}{2})\ +\ f'(\overline{b}\,)} \tag{A10}$$

Finally, to further correct the predicted (corrected) value, we use approximation, $\overline{\overline{b}}$, as the iterate value instead of Newton–Raphson techniques value, a; we have

$$\overline{\overline{\overline{b}}} \cong \overline{\overline{b}}\ -\ \frac{f(\overline{\overline{b}})}{f'(\overline{\overline{b}})}$$

In Summary, we have

Step 1 Newton–Raphson

$$\overline{b}\ \cong a -\ \frac{f(a)}{f'(a)} \tag{A11}$$

Step 2 (Midpoint and Simpson p = 1/2)

$$\overline{\overline{b}} \cong\ a -\ \frac{6\ f(a)}{pf'(a)\ +\ (6\ -\ 2p)f'(\frac{a\ +\ \overline{b}}{2}) + p\ f'(\overline{b}\,)} \tag{A12}$$

Step 3 (Modified Newton–Raphson)

$$\overline{\overline{\overline{b}}} \cong \overline{\overline{b}} - \frac{f(\overline{\overline{b}})}{f'(\overline{\overline{b}})} \tag{A13}$$

Algorithm (Oghovese–John)

Let f, x_0 be given, Newton–Raphson approximation iterates become

$$x_{n+1}\ \cong\ xn\ -\ \frac{f(x_n)}{f'(x_n)} \tag{A14}$$

Initialize $u_0 = v_0 = x_0$.

For n = 0: maxIteration

// for n = 0, x_n, u_n, v_n are known from

$//u_0 = x_0 - \frac{f(x_0)}{f'(x_0)}$

$//v_0 = u_0 - \frac{f(u_0)}{f'(u_0)}$

$$u_{n+1} \cong xn - \frac{f(x_n)}{f'(x_n)} \tag{A15}$$

//using this we get

$$v_{n+1} \cong v_n - \frac{12\, f(v_n)}{f'(v_n) + 10\, f'(\frac{v_n + u_{n+1}}{2}) + f'(u_{n+1})} \tag{A16}$$

// finally

$$x_{n+1} = v_{n+1} - \frac{f(v_{n+1})}{f'(v_{n+1})} \quad n \geq 0 \tag{A17}$$

// we get this x_{n+1} instead of $x_{n+1} \cong x_n - \frac{f(x_n)}{f'(x_n)}$

//break when x_{n+1} is acceptable using

// Tolerance ε = 0.0000001, maxIterations = 100

// Iterations terminate when $(|x_{n+1} - x_n| + |f(x_{n+1})|) < \varepsilon$ or maxIteration reached

EndFor

Appendix B

The algorithm is implemented in Matlab using the hybrid of three methods.

```
function [iter,root,roots,ea,bl,br] = hybridN(f, df, a, b, es, imax)
%{
        input:
  f        the function
  a, xl    lower value bracket
  b, xu    upper value bracket
  es       error stopping critia
  imax     upper bound on the number of iterations
        output:
  iter     the number of iterations
  root     approxmate final root
  roots    approxmate iterated rootss
  ea       error at each iteration
  bl       lower value brack at each iteration
  br       upper value brack at each iteration
        Evaluation for bisection, false position and for newtons methods for evaluation
with additional documention is standard call to keep the hybrid code simple
            [iter,root,roots,ea,bl,br] = falsePos(f, a,b, es, 1);
            [iter,root,roots,ea,bl,br] = bisection(f, a,b, es, 1);
                  [iter,root,roots,ea] = newton(f, a, es, 1);
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%
%%%%%%%%%%
        % Initializaation
            xl = a; xu = b; bl(1) = xl; br(1) = xu;
            rold = xl;
            ea(1) = 0;
            root = 0;
            roots = 0;
            iter = 0;
        if f(a)*f(b) > 0 % if guesses do not bracket for Bisection,False Postion methods
                    error('Root not Bracketed')
            return
        end
        % iterations begin here
        for i = 1:imax
            iter = i;
                % first use bisction and false position predicted point
                [iterb,rootb,rootsb,eab,blb,brb] = bisection(f, xl, xu, es, 1);
```

```
[iterse,rootse,rootsse,ease,blse,brse] = falsePos(f, xl, xu, es, 1);

if (abs(f(rootb)) < abs(f(rootse)))
    root = rootb;
  else
    root = rootse;
 end
  xl = max(blb(1), blse(1)); xu = min(brb(1), brse(1));
  % then use newton predicted point
  [itern,rootn,rootsn,ean] = newton(f,df,(xl,es,1);
  if (abs(f(rootn)) < min(abs(f(xl)), abs(f(xu))))
        if(f(rootn)*f(xu) < 0))
          if (xl < rootn) && (xu > rootn)
              xl = rootn;
          end
        else
              if (xl < rootn) && (xu > rootn)
              xu = rootn;
               end
          end
          root = rootn;
        end
    % for documentation
          bl(i) = xl; br(i) = xu;
          r = root;
          ea(i) = abs(f(root)) + abs(r − rold); % absolute error
        roots(i) = root;
        if ea(i) < es
          break;
        end
    % for next iteration
        rold = root;
end
root = r;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

References

1. Dekker, T.J.; Hoffmann, W. *Algol 60 Procedures in Numerical Algebra, Parts 1 and 2*; Tracts 22 and 23; Mathematisch Centrum: Amsterdam, The Netherlands, 1968.
2. Dekker, T.J. Finding a zero by means of successive linear interpolation. In *Constructive Aspects of the Fundamental Theorem of Algebra*; Dejon, B., Henrici, P., Eds.; Wiley-Interscience: London, UK, 1969; ISBN 978-0-471-20300-1. Available online: https://en.wikipedia.org/wiki/Brent%27s_method#Dekker\T1\textquoterights_method (accessed on 3 March 2021).
3. Brent's Method. Available online: https://en.wikipedia.org/wiki/Brent%27s_method#Dekker\T1\textquoterights_method (accessed on 12 December 2019).
4. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. Van Wijngaarden–Dekker–Brent Method. In *Numerical Recipes: The Art of Scientific Computing*, 3rd ed.; Cambridge University Press: New York, NY, USA, 2007; ISBN 978-0-521-88068-8.
5. Sivanandam, S.; Deepa, S. Genetic algorithm implementation using matlab. In *Introduction to Genetic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 211–262. [CrossRef]
6. Moazzam, G.; Chakraborty, A.; Bhuiyan, A. A robust method for solving transcendental equations. *Int. J. Comput. Sci. Issues* **2012**, *9*, 413–419.
7. Oghovese, O.; John, E.O. New Three-Steps Iterative Method for Solving Nonlinear Equations. *IOSR J. Math. (IOSR-JM)* **2014**, *10*, 44–47. [CrossRef]
8. Fitriyani, N.; Imran, M.; Syamsudhuha, A. Three-Step Iterative Method to Solve a Nonlinear Equation via an Undetermined Coefficient Method. *Glob. J. Pure Appl. Math.* **2018**, *14*, 1425–1435.
9. Darvishi, M.T.; Barati, A. A third-oredr Newton-type method to solve system of nonlinear equations. *Appl. Math. Comput.* **2007**, *187*, 630–635.

10. Weerakoon, S.; Fernando, T.G.I. A variant of Newton's method with accelerated third-order convergence. *Appl. Math. Lett.* **2000**, *13*, 87–93. [CrossRef]
11. Khattri, S.K.; Abbasbandy, S. Optimal Fourth Order Family of Iterative Methods. *Mat. Vesn.* **2011**, *63*, 67–72.
12. Ostrowski, A.M. *Solutions of Equations and System of Equations*; Academic Press: New York, NY, USA; London, UK, 1966.
13. Cordero, A.; Torregrosa, J.R. Variants of Newton's Method using fifth-order quadrature formulas. *Appl. Math. Comput.* **2007**, *190*, 686–698. [CrossRef]
14. Jayakumar, J. Generalized Simpson-Newton's Method for Solving Nonlinear Equations with Cubic Convergence. *IOSR J. Math. (IOSR-JM)* **2013**, *7*, 58–61. [CrossRef]
15. Sabharwal, C.L. Blended Root Finding Algorithm Outperforms Bisection and Regula Falsi Algorithms. *Mathematics* **2019**, *7*, 1118. [CrossRef]
16. Chun, C. Iterative method improving Newton's method by the decomposition method. *Comput. Math. Appl.* **2005**, *50*, 1559–1568. [CrossRef]
17. Hasanov, V.I.; Ivanov, I.G.; Nedjibov, G. A new modification of Newton's method. *Appl. Math. Eng.* **2002**, *27*, 278–286.
18. Singh, M.K.; Singh, A.K. A Six-Order Method for Non-Linear Equations to Find Roots. *Int. J. Adv. Eng. Res. Dev.* **2017**, *4*, 587–592.
19. Osinuga1, I.A.; Yusuff, S.O. Quadrature based Broyden-like method for systems of nonlinear equations. *Stat. Optim. Inf. Comput.* **2018**, *6*, 130–138. [CrossRef]
20. Babajee, D.K.R.; Dauhoo, M.Z. An analysis of the properties of the variants of Newton's method with third order convergence. *Appl. Math. Comput.* **2006**, *183*, 659–684. [CrossRef]
21. Grau, M.; Diaz-Barrero, J.L. An Improvement to Ostrowski Root-Finding Method. *Appl. Math. Comput.* **2006**, *173*, 450–456. [CrossRef]
22. Sharma, J.R.; Guha, R.K. A Family of Modified Ostrowski Methods with Accelerated Sixth Order Convergence. *Appl. Math. Comput.* **2007**, *190*, 111–115. [CrossRef]
23. Fang, L.; Chen, T.; Tian, L.; Sun, L.; Chen, B. A Modified Newton-Type Method with Sixth-Order Convergence for Solving Nonlinear Equations. *Procedia Eng.* **2011**, *15*, 3124–3128. [CrossRef]
24. Halley at Wikipedia. Available online: https://en.wikipedia.org/wiki/Halley%27s_method#Cubic_convergence (accessed on 22 December 2019).
25. Newton. *Philosophiae Naturalis Principia Mathematica*; Sumptibus Cl. et Ant. Philibert: Geneva, Switzerland, 1760.
26. Ortega, J.M.; Rheinboldt, W.C. *Iterative Solutions of Nonlinear Equations in Several Variables*; Academic Press: New York, NY, USA, 1970.
27. Santra, S.S.; Ghosh, T.; Bazighifan, O. Explicit criteria for the oscillation of second-order differential equations with several sub-linear neutral coefficients. *Adv. Diff. Equ.* **2020**, *2020*, 643. [CrossRef]